

Report No. GTRI-TR-98-A5347-F

FEDERATION TESTING PROCESS AND TOOLS

Federation Test and Management Integration Support & Study (FTAM ISS) Final Results Report

CDRL A002

April 30, 1998

Prepared For:

STRICOM

12350 Research Parkway

Orlando, FL 32826

Contract Number: N61339-97-K-0001

Prepared By:

David W. Roberts, Margaret M. Horst, J. Andrew Old

Distributed Simulation Systems Group

Computer Science and Information Technology Division

Information Technology & Telecommunications Laboratory

GEORGIA TECH RESEARCH INSTITUTE

GEORGIA INSTITUTE OF TECHNOLOGY

A UNIT OF THE UNIVERSITY SYSTEM OF GEORGIA

ATLANTA, GEORGIA 30332-0800

This Page Intentionally Left Blank

FOREWORD

This report was prepared by the FTAM ISS Team consisting of David W. Roberts (Principal Investigator), Margaret Horst, and J. Andrew Old. Significant contributions to the success of this project and to the final report were made by the members of the Distributed Simulation Systems (DSS) Group, especially Margaret Loper, Thom McLean, Kyle Crawford, Laura Burkhart, and Debbie Esslinger. Very special thanks go to Amanda Crowell for her tireless effort on editing the many versions of this final report. Thanks also go to Terry Hilderbrand, the Computer Science and Information Technology Division Chief, and Randolph Case, the Information Technology and Telecommunications Lab Director, for their management guidance and oversight. Thanks also go to the MAPS team, especially Mary McKenna, for their timely support.

The GTRI FTAM ISS team worked in support of the Federation Test System (FTS) Integrated Development Team (IDT). The FTS IDT was made up of STRICOM representatives and contractors from TASC and Acusoft. Many of the ideas expressed in this report were either authored, revised, or significantly influenced by the members of the IDT, including: (STRICOM) Rodney Long, Tom Verscharen, Bernie Gajkowski; (Acusoft) Kevin Mullally, Jaime Cisneros; and (TASC) Frank Valdes.

The FTS IDT also participated actively in the Simulation Interoperability Standards Organization (SISO) in an effort to understand the broad simulation testing requirements of the community and to publish the results of our research and development efforts to as wide an audience as possible. In support of this goal, David Roberts and Kevin Mullally served as Chair of the Simulation Interoperability Workshop (SIW) Testing Forum. Many improvements have been made to the process and tools that have been developed in the IDT thanks to the feedback of the Testing Forum.

This Page Intentionally Left Blank

TABLE OF CONTENTS

1. SCOPE.....	1
2. INTRODUCTION.....	3
2.1 The Requirement for HLA	3
2.2 Transition from DIS to HLA Standards	3
3. FEDERATION TESTING SYSTEM (FTS) INTEGRATED DEVELOPMENT TEAM (IDT).....	5
3.1 Background	5
3.2 Charter	5
3.3 Development Process	5
3.3.1 Requirements Development	5
3.3.1.1 IDT 1	6
3.3.1.2 IDT 2	6
3.3.1.3 IDT 3	7
3.3.2 FTS Design Meetings.....	8
3.3.3 Implementation and Testing	8
3.4 Leveraged Support of AMG Working Groups and SIW User Forums.....	8
3.5 Web Site for Simulation Testing	9
4. FEDERATION TESTING PROCESS.....	11
4.1 HLA Guideline Processes	11
4.1.1 HLA Federation Development and Execution Process (FEDEP)	11
4.1.1.1 FEDEP Version 1.1	11
4.1.1.2 Federation Testing within the FEDEP.....	13
4.1.1.3 RTI Initialization Data (RID).....	14
4.1.1.4 Federation Execution Details (FED)	14
4.1.1.5 Federation Execution Planning Workbook (FPW)	14
4.1.2 HLA Compliance Testing	14
4.2 Federation Testing Process	16
4.2.1 Application Testing	18
4.2.1.1 Definition	18
4.2.1.2 Discussion.....	19
4.2.2 Integration Testing.....	20

4.2.2.1 Definition	20
4.2.2.2 Discussion	20
4.2.3 Functional Testing	22
4.2.3.1 Definition	22
4.2.3.2 Discussion	23
4.2.4 Scenario Testing	24
4.2.4.1 Definition	24
4.2.4.2 Discussion	24
4.2.5 Other Types of Testing Within the Federation Testing Process	26
4.2.5.1 Unit Testing	26
4.2.5.2 Exception Testing	27
4.2.5.3 Performance Testing	27
4.2.6 Automation of the End to End Federation Testing Process	27
4.3 Relationship between Existing Processes and the Federation Testing Process	28
4.3.1 Relationship between Federation Testing and the HLA FEDEP	29
4.3.2 Relationship between Federation Testing & VV&A Processes	30
4.3.3 Application of the Federation Testing Process to Federations	32
5. DATA AND METHOD REQUIREMENTS FOR FEDERATION TESTING	34
5.1 Data Requirements for Federation Testing	34
5.1.1 Federation Object Models (FOMs)	34
5.1.1.1 Central FOM Library	35
5.1.1.2 RPR FOM	37
5.1.1.3 Relevance to Federation Testing	37
5.1.2 Object Interaction Protocols	38
5.1.2.1 Background	38
5.1.2.2 OIPs for Federation Testing	38
5.1.2.3 Describing and Documenting OIPs	39
5.1.2.4 Relevance to Federation Testing	39
5.1.3 Federation Agreements	41
5.1.3.1 Overview	43
5.1.3.2 Current State	45
5.1.3.3 Relevance to Federation Testing	45

5.1.4	Fedex Planning Workbook (FPW)	46
5.1.4.1	Description	46
5.1.4.2	Relevance to Federation Testing.....	48
5.2	Method Requirements for Federation Testing.....	48
5.2.1	Use of the Management Object Model (MOM)	49
5.2.1.1	MOM and Federate Compliance Testing.....	49
5.2.1.2	MOM and Federation Testing.....	50
5.2.2	File Formats	51
5.2.2.1	ModSAF RDR.....	51
5.2.2.2	JMASS File Formats.....	51
5.2.2.3	Data Interchange Formats (DIFs).....	53
5.2.2.4	Self-Describing Data Interchange Formats (DIFs).....	55
5.2.3	Test Procedures	56
5.2.3.1	Test Procedures Format.....	56
5.2.3.2	Test Procedures Description	58
5.2.3.3	Test Procedures Example	58
5.2.4	Integration of HLA Tools Developed By Different Organizations	58
	Object Model Development Tool (OMDT)	59
5.2.4.2	FPW Development Tool	60
5.2.5	Use of the Internet to Facilitate Test Management.....	60
5.2.5.1	Federate Conformance Testing Example	60
5.2.5.2	Implications for Federation Testing and Future Directions	62
6.	TESTING TOOLS	64
6.1	Testing Tools for Distributed Simulation.....	64
6.2	HLA Compliance Test Tools.....	66
6.2.1	Pre-Processor	66
6.2.1.1	Conformance Cross-Checker	66
6.2.1.2	Nominal Sequence Generator	66
6.2.1.3	RepSOM Generator	66
6.2.1.4	Test Sequence Generator	67
6.2.2	Logger	67
6.2.3	Post-Processor.....	67
6.3	Federation Test System (FTS)	67

6.3.1	Test Federate	69
6.3.2	Analysis Federate.....	70
6.3.3	Test Session Manager	70
6.3.4	Exercise Support Tools	71
6.3.5	Next Steps for the FTS.....	72
7.	RECOMMENDATIONS.....	74
7.1	Policy Recommendations.....	74
7.1.1	Support Development of a DIS Federation.....	74
7.1.2	Continue Support of the SIW Testing Forum.....	75
7.2	Technical Recommendations	75
7.2.1	Encourage Expansion of the FEDEP to Describe Federation Testing in More Detail.....	75
7.2.2	Promote the Development and Use of Interchange Formats for HLA Data	75
7.2.2.1	FPW	76
7.2.2.2	MSCs for OIPs	76
7.2.2.3	Test Procedures.....	76
7.2.2.4	Federation Agreements	76
7.2.3	Use the Internet to Facilitate Test Management.....	77
7.2.4	Encourage Applied Research to Investigate Functional and Scenario Testing Requirements.....	77
7.2.5	Encourage Basic Research to Develop Approaches for the End-to- End Federation Testing Automation.....	77
7.2.6	Promote the Practice of Providing Test Procedures with Standards Proposals.....	78
7.2.7	Leverage Testing Research and Practice	78
7.3	FTS Development Recommendations.....	78
7.3.1	Use the FTS to Support Federate and Tool Development.....	79
7.3.2	Use the FTS to Support Federation Development.....	80
7.3.3	Develop Integration, Functional, and Scenario Testing Capabilities..	80
7.4	Timeline.....	81
8.	REFERENCES.....	82
9.	APPENDIX.....	84
9.1	ModSAF RDR.....	84

9.2	Message Sequence Chart (MSC) Depicting a Fire and Detonate Series between Two Federates.....	87
9.3	FTS Test Procedures Example	90
9.4	Distributed Simulation Systems (DSS) Lab	92
9.4.1	DSS Lab Environment Setup for Federation Tests.....	92
9.4.1.1	Multiple RTI Versions	92
9.4.1.2	DIS to HLA Migration.....	93
9.4.1.3	JMASS and Threat Radar Simulation Federates Used in IRAD Project 93	
9.4.2	HLA Distributed Simulation Interface Framework (DSIF)	94
9.4.2.1	Current Status	95
9.4.2.2	Relevance to Federation Testing.....	95
9.5	Acronym List.....	96

1. SCOPE

This report documents research supporting the US Army Simulation, Training and Instrumentation Command (STRICOM) on the Federation Testing and Management Integration Support and Study (FTAM ISS) BAA contract. The contract covered the period from December 3, 1996 to April 30, 1998. The purpose of the project was to support STRICOM in the understanding of the requirements for HLA Testing, and to provide support to the design and development of an HLA Federation Test tool.

There are two main goals of this report:

- to document the research accomplished in support of the Federation Test System (FTS) Integrated Development Team (IDT) in process development and FTS requirements and development support
- to document the history and current status of Federation Testing, as well as the future requirements and necessary research areas for the FTS team, STRICOM, and the distributed simulation community.

The report is organized into the following sections:

1. SCOPE: Describes the purpose and contents of this Final Report
2. INTRODUCTION: Introduces the reader to the context of the report. Introduces the HLA and the issues involved with distributed simulation testing.
3. FEDERATION TESTING SYSTEM (FTS) INTEGRATED DEVELOPMENT TEAM (IDT) : Describes the process that was used to define the requirements for HLA Testing and to design the FTS.
4. FEDERATION TESTING PROCESS: Describes two guideline processes for the HLA: the Federation Development and Execution Process (FEDEP) and the HLA Compliance Testing process. Proposes a Federation Testing Process, compares it to existing distributed simulation processes, and describes an approach to automating the end-to-end process.
5. DATA AND METHOD REQUIREMENTS FOR FEDERATION TESTING: Describes data requirements for Federation Testing, including Federation Object Models (FOMs), Object Interaction Protocols, Federation Agreements, and the Fedex Planning Workbook (FPW). Describes method requirements for Federation Testing, including the use of the Management Object Model

(MOM), standardized file formats, standardized test procedures, the internet, and integrated tools.

6. TESTING TOOLS: Describes test tools for distributed simulation, including "legacy" test tools, HLA Federate Compliance Test Tools, and the Federation Test System (FTS).
7. RECOMMENDATIONS: Describes recommendations from this research and development effort, including policy recommendations, technical recommendations, and FTS development recommendations. Presents a timeline for context.
8. REFERENCES: Complete bibliographical list of documents referenced in this report.
9. APPENDIX: Provides more detail on file formats, test procedures, the DSS lab environment, the Distributed Simulation Interface Framework (DSIF), and acronym list.

2. INTRODUCTION

The research described in this report was conducted to develop concepts, processes, requirements, and tools to support HLA simulation testing. This introduction section describes the context in which this research was conducted by describing the HLA and its importance for the distributed simulation community.

2.1 THE REQUIREMENT FOR HLA

In accordance with the Department of Defense (DoD) Modeling and Simulation Master Plan, the Defense Modeling and Simulation Office (DMSO) led a DoD-wide effort to create a common technical framework. The goal of this framework is to facilitate the interoperability of models and simulations (M&S) among themselves and with Command, Control, Communications, Computers, and Intelligence (C4I) systems, as well as to promote the reuse of M&S components. The HLA Baseline Definition was approved by the Under Secretary of Defense for Acquisition and Technology [USD(A&T)] on 10 September 1996.

The memorandum signed by Dr. Paul Kaminski, USD(A&T), requires that all new DoD simulation programs use the HLA and sets a timetable for the review of simulations and development of migration plans for existing simulations. As of FY99, DoD will provide no further funding for the development of non-compliant simulations, and as of FY01, non-compliant simulations may not be used on DoD projects.

In December 1997, HLA was accepted by the Executive Committee of the Simulation Interoperability Standards Organization (SISO) and approved by the IEEE for development as an IEEE standard. The standards are slated to be:

- Framework and Rules (1516)
- Federate Interface Specification (1516.1)
- Object Model Template Specification (1516.2).

With this step, the HLA is progressing from being a US DoD standard to an international draft standard.

2.2 TRANSITION FROM DIS TO HLA STANDARDS

With the adoption of the HLA as a DoD mandate and as a future IEEE standard, many research efforts are being conducted to understand what is required to transition legacy simulations to the HLA and to develop new approaches to distributed simulation development that will take advantage of the HLA

architecture. However, although many issues have changed with the advent of the HLA, many have remained the same. These include:

- Bringing together disparate simulations and integrating them into a system that supports the needs of the user is still a complex task.
- Detailed processes are still needed for the development of federations of simulations.
- Obtaining interoperability between distributed simulations is still a system engineering task that has to be addressed every time distributed simulations are brought together for the first time or for a new purpose.
- *Testing* is needed to verify that simulations can be used for the intended purpose.
- *Test processes* are needed to guide the development of federations of simulations.
- *Automated tools* are needed to support the test process.

This report provides more details on these issues.

3. FEDERATION TESTING SYSTEM (FTS) INTEGRATED DEVELOPMENT TEAM (IDT)

The purpose of this section is to document the evolutionary process that the Federation Testing System (FTS) Integrated Development Team (IDT) went through to develop a Federation Test Process and the FTS.

3.1 BACKGROUND

The IDT was formed in early February 1997 to tackle the complex issues associated with the understanding of Federation Testing and the evolution of existing Distributed Interactive Simulation (DIS) test tools to HLA test tools. The group was made up of members from STRICOM, TASC, AcuSoft, and GTRI.

3.2 CHARTER

One of the first steps the IDT took was to draft a Concept of Operation (COO) to outline its responsibilities, makeup, and products. The following goals were developed (taken from the IDT Charter):

1. Document user and technical requirements for Federation Testing.
2. Document user and technical requirements for an FTS.
3. Leverage existing STRICOM development efforts (e.g. DTS, DISECT).
4. Capture and develop expertise among group members in all areas and issues of Federation Testing.
5. Stay abreast of developing HLA technology issues and thrusts.
6. Provide feedback to the modeling and simulation (M&S) community on issues related to federation development, including details on the process that should be used for Federation Testing.

The IDT's objectives included developing the FTS and integrating it into the wider HLA community. Thus, the emphasis was on supporting more than DIS and its transition to HLA. The IDT focused on developing a process and tool that would support other federations, as well.

3.3 DEVELOPMENT PROCESS

As referenced above, the development process was composed of three phases: requirements, implementation, and testing.

3.3.1 Requirements Development

The first phase of the IDT process was to define the requirements for Federation Testing and the development of a Federation Testing capability. This goal was

accomplished during several IDT meetings, which are described below to give the reader an understanding of the steps taken to understand, define, and refine the problem.

IDT meetings were held regularly to discuss the issues related to Federation Testing and to develop strategies to accomplish the goals referenced in the previous section. The following sections summarize the issues and discussions of the first few critical IDT meetings.¹

3.3.1.1 IDT 1

The first IDT meeting held on February 14, 1997 served as an introductory meeting for members to get to know each other and begin developing the Concept of Operation. Since the team members recognized early on that this process would be a migration from the existing DIS test tools, a briefing on the DIS Test Suite (DTS), its test process, and DISECT was given. Members also discussed HLA standards to bring everyone up to speed on the issues related to evolving from the DIS Architecture to the HLA. Finally, the members made an initial attempt to decide what to test, with emphasis placed on designing the tests in a way that would support other federation needs, not just those of the DIS. They concluded that four areas needed to be addressed:

1. Federation Object Model (FOM) and Simulation Object Model (SOM) objects and attributes
2. Object Interaction Protocols (OIPs)
3. Federation Agreements
4. Federation Required Execution Details (FRED)

The next step was to define the above areas, specifically the items that were not well defined in the HLA documents: OIPs, Federation Agreements, and FRED.

3.3.1.2 IDT 2

The second IDT meeting was held on March 11, 1997. As an extension of the first meeting, the focus was on clarifying the testing areas referenced above. First, team members developed a sample OIP depicting a detect-and-destroy sequence involving two tanks using a Message Sequence Chart (MSC) (See Section 5.1.2 for more information on OIPs.). They also attempted to formalize the FRED and eventually decided that three areas would be tested to meet federation requirements: FOM/SOM objects and attributes, Federation Agreements, and the FRED. With the testing areas agreed upon, the next task

¹ The sections describing the IDTs were taken directly from the minutes of the IDT meetings.

was to define the process more concretely, including developing examples for what would occur during each step of the process. Members also raised the issue of what role the federate Capability Statement should play. They concluded that the capability statement is more than just a SOM, but they were still unclear on its exact format. Some time was spent developing a capability statement.

3.3.1.3 IDT 3

The focus of the third IDT meeting was to discuss the HLA testing process and begin discussing the FTS design. The HLA testing process consists of two main parts: Compliance Testing (also called Federate Testing) and Federation Testing. There are three subparts to Compliance Testing (originally taken from the HLA Rules (DMSO, 1997e)):

- Federate compliance. The federate must conform to the following six federate rules:
 1. Provide HLA SOM in Object Model Template (OMT) format.
 2. Update/Reflect Objects & Send/Receive Interactions.
 3. Dynamically transfer and/or accept ownership of attributes.
 4. Vary thresholds of attributes.
 5. Manage local time to coordinate data exchange with federation members.
 6. Interact with Runtime Infrastructure (RTI) according to Interface Specification (IFSpec).
- Federation compliance. The federation must conform to the following federation rules:
 1. HLA FOM in the OMT format
 2. Object representations in federates, not RTI.
 3. All runtime exchange of FOM data through RTI.
 4. Federations interact with RTI according to IFSpec.
 5. During execution, an attribute of an instance of an object shall be owned by only one federate at a time.
- RTI compliance. The aspects of RTI compliance are as follows:
 1. RTI shall interact with federations according to the IFSpec.

2. RTI provides Interface services to federates according to the RTI functional specifications.
3. Object representations are in federates, not RTI.
4. RTI shall enforce the rule that an attribute of an object can only be owned by one federate at a time.

In addition to Compliance Testing, there are four parts of Federation Testing: Application Testing, Integration Testing, Functional Testing, and Scenario Testing. The focus of the first phase of FTS development was to define and develop the testing method for Application Testing.

As a result of several teleconferences, an initial chart was developed on the components involved in Application Testing (See Section 6 for the results of the multiple iterations of this diagram). The next step was to determine what would be developed as part of the initial FTS prototype. Also at issue was the process of test procedure development.

3.3.2 FTS Design Meetings

Once the basic concept for the FTS was established, the next step was formalizing the design requirements for the system. The initial results of the design meetings were two documents: the FTS Application Testing Process Requirements Definition and the FTS Application Testing Process System/Subsystem Design Description (S/SDD) (AcuSoft, 1997a) (AcuSoft, 1997b). These documents define the components of the FTS and its general functionality, which the IDT has determined will be important to federation developers.

3.3.3 Implementation and Testing

Currently, the IDT is involved in the implementation and testing phases of FTS development. Currently the FTS developer is working with the Real-Time Reference FOM (RPR FOM), but, note that the FTS is designed to work with any FOM. Scripts are being written manually to control the actions of the test federate, the federate under test (FUT), and the analysis federate. The focus of the current effort is to develop a large number of scripts in support of the RPR FOM, to serve as a "proof-of-principle" prototype in the development process. Section 6 presents the results of the initial implementation efforts.

3.4 LEVERAGED SUPPORT OF AMG WORKING GROUPS AND SIW USER FORUMS

DMSO sponsors several working groups, which report to the Architecture Management Group (AMG) about specific technical areas in the HLA. These working groups are managed by the DMSO Chief Scientist, Dr. Judith Dahmann, and GTRI leads and/or participates in four such groups. The technical areas of

these groups are mirrored in some of the User Forums at the Simulation Interoperability Workshop (SIW) sponsored by the Simulation Interoperability Standards Organization (SISO). GTRI also actively participates in the User Forums, and several GTRI personnel have been elected to Paper Review Panels for some Forums.

While the STRICOM contract with GTRI did not directly support these activities, GTRI was able to leverage these efforts to provide updated information to the FTS IDT in important technical areas. These four technical areas -- federation development and object model tools, federation management and Management Object Model (MOM), RTI performance, and Federate Testing -- are discussed in more detail in later sections.

3.5 WEB SITE FOR SIMULATION TESTING

As part of an effort to support users of both the DIS Test Suite (DTS) and the FTS, a World Wide Web site has been established at <<http://www.ads-test.org>>. This site is intended to serve as a repository for simulation testing documents and software and to provide a way to collect administrative and technical data relevant to STRICOM's testing activities.

In support of GTRI's efforts under the Federation Test and Management contract, a Web site also was developed to serve as a common area for interested parties within GTRI's Distributed Simulation Systems (DSS) group to obtain current and relevant information on the FTAM project. This Web site contains a concise summary of the purpose and progress made through the efforts of the FTS-IDT, as well as relevant links and papers.

To best leverage the information in both Web sites, the best of the GTRI site was integrated into the ADS-Test Site.

This Page Intentionally Left Blank

4. FEDERATION TESTING PROCESS

One of the goals of the FTAM project was to define a Federation Testing Process that could be used to guide development of the FTS. This section describes the proposed Federation Testing Process, describes the HLA processes within which the Federation Test Process must fit, and compares the developed process with existing distributed simulation processes. The HLA Federation Development and Execution Process (FEDEP) and the HLA Conformance Testing Process are described first to provide the context necessary for the Federation Testing Process discussion.

4.1 HLA GUIDELINE PROCESSES

This section describes the HLA Federation Development and Execution Process (FEDEP) and the HLA Conformance Testing Process. Over the course of the FTAM program, these processes and supporting tools have gone through several revisions. The versions described here are the latest releases of these processes.

4.1.1 HLA Federation Development and Execution Process (FEDEP)

The HLA Federation Development and Execution Process (FEDEP) is intended to "...identify and describe the sequence of activities necessary to construct HLA federations." ² The baseline version of the document is Version 1.0, dated 6 September 1996. (DMSO, 1996c)

4.1.1.1 FEDEP Version 1.1

At the Fall 1997 Simulation Interoperability Workshop (SIW), the Federation Development Process User Forum (PROC) presented papers intended to further clarify the FEDEP. Comments from the audience indicated a general feeling that the entire process was still too complicated and cumbersome and that significant improvements were needed in the presentation of the process. In response to these comments, the Federation Development Working Group has published an updated version of the FEDEP: Version 1.1, dated 9 December 1997. (DMSO, 1997b) Both documents can be found at the DMSO HLA web site. <<http://hla.dmsomil>>

² In addition to the processes discussed in this section, it is also important to note the existence of the DIS Exercise Management & Feedback (EMF) Recommended Practice document, which was approved as IEEE Standard 1278.3. (IEEE, 1997a) This document establishes the EMF requirements for participation in a DIS exercise and may also be useful in understanding the steps that need to be taken to create a distributed simulation system.

Version 1.1 of the FEDEP presents a five-step federation development process:

- **Requirements Definition:** The federation sponsor and federation development team must define and agree on a set of objectives, then document what must be accomplished to achieve those objectives.
- **Conceptual Model Development:** A representation of the real-world domain of interest (entities and tasks) is developed and described in terms of a set of required objects and interactions.
- **Federation Design:** Federation participants are determined (if not previously identified), and a FOM is developed to explicitly document information exchange requirements and responsibilities.
- **Federation Integration and Test:** All necessary federation implementation activities are performed, and testing is conducted to ensure interoperability requirements are being met.
- **Execution and Analysis Results:** The federation is executed, outputs are analyzed, and feedback is provided to the federation sponsor.

Figure 4.1 shows the latest version of the FEDEP.

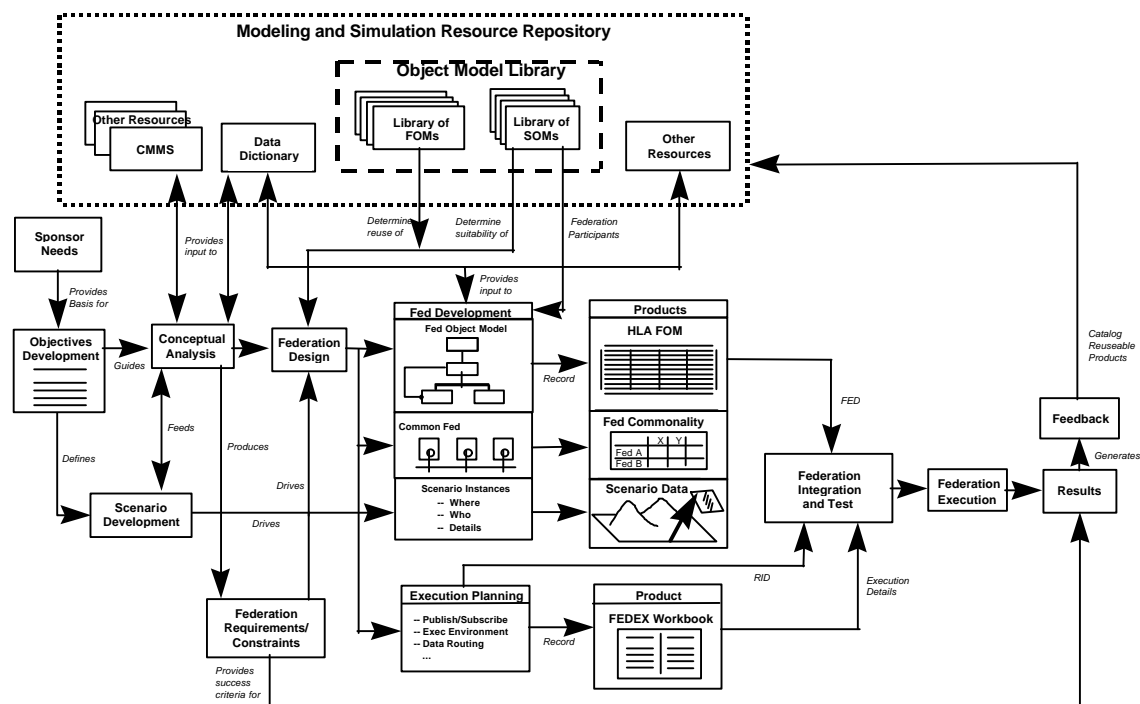


Figure 4.1. Federation Development and Execution Process Model

While the figure resembles prior FEDEP charts, there are some important refinements, particularly related to the Model and Simulation Resource Repository (MSRR), that reflect an increasing understanding of the practical ways in which federations must be developed. The boxes in the FEDEP chart can be mapped to the five-step process as indicated in Table 4.1 below

Requirements Definition	Conceptual Model Development	Federation Design	Federation Integration and Test	Execute and Analyze Results
Sponsor Needs Identification	Scenario Development	Federation Design	Execution Planning	Federation Execution
Objectives Development	Conceptual Analysis	Federation Development	Federation Integration and Test	Results and Feedback

Table 4.1. Mapping of FEDEP to Five-Step Process

4.1.1.2 Federation Testing within the FEDEP

Figure 4.2 below shows the portion of the FEDEP describing Federation Testing.

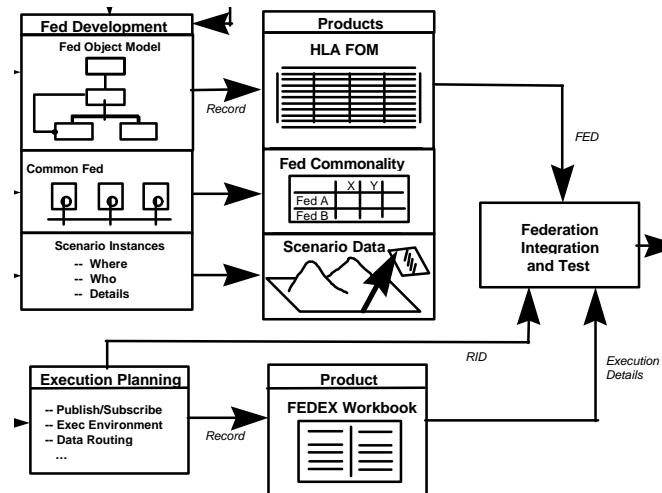


Figure 4.2 Testing Within the FEDEP

The first step to creating a Federation Testing process is to see how it fits into the overall FEDEP. Note that the current FEDEP defines several sources of

documented requirements that drive the Federation Testing process: the Runtime Infrastructure (RTI) Initialization Data (RID), the Federation Execution Details (FED), and the Federation Execution Planning Workbook (FPW).

4.1.1.3 RTI Initialization Data (RID)

The RID is the data needed by the RTI to operate -- the RTI's configuration file. (DMSO, 1997c) It specifies the number of times the RTI should tick, what port should be used to communicate, and the name of the machine on which the RTI exec is running. Without a properly formatted RID file, the RTI will not function properly. All tests that involve two or more federates must have a correct RID file to start.

4.1.1.4 Federation Execution Details (FED)

The FED file is a text file representation of the Federation Object Model (FOM) which is read into the RTI and federates. It specifies object classes, interactions, attributes, and parameters. It also is formatted to show inheritance between classes. The RTI uses the FED file to assign handles to objects. Thus, it is important that federates within a federation have consistent FED files. Problems with the FED file are often discovered during the initial phases of integration and testing.

4.1.1.5 Federation Execution Planning Workbook (FPW)

The FPW is a replacement of the earlier concept of the Federation Required Execution Details (FRED). The FPW describes general federation operating and performance characteristics. Further work is needed in defining the FPW, including the development of a file format and tool to facilitate creation and management of the FPW. The FPW is described in detail in Section 5.1.4 in the context of additional requirements for the automation of federation testing.

4.1.2 HLA Compliance Testing ³

The HLA Compliance Process has two parts, Conformance and Certification. Conformance is the process of verifying that an implementation performs in accordance with a particular standard (Knightson, 1993). For HLA, Conformance is testing a federate to the Interface Specification (IFSpec) and Object Model Template (OMT) standards, per the HLA Compliance Checklist (DMSO, 1996d). Certification is the process of validating that an implementation has been tested

³ This section was adapted directly from the series of Compliance Testing papers published by GTRI at SIW and DIS workshops and the "Federate Test Tools Operator Guide, Version 1.1," prepared for DMSO (DMSO, 1997a).

for Conformance. This means that once a federate under test (FUT) has completed Conformance Testing, the results must be validated before being recorded in a "certified products list" (e.g., the MSRR). Certification will be executed by a Certification Agent (CA) who is accredited by the government.

For HLA compliance, the standards tested are the IFSpec and the OMT. In accordance with Federate Compliance Checklist item 1, a federate must have a Simulation Object Model (SOM) in the OMT format. The SOM Conformance Test ensures that SOM data is consistent across tables. This test has three checks: Parseability, Completeness, and Consistency. The services asserted in the FUT's Capability Statement (CS) also are cross-checked against services implied by the FUT's SOM to determine if the two specifications are consistent.

In accordance with Federate Compliance Checklist items 2-6, a federate must be capable of supporting the services in the IFSpec, as required by the capabilities specified in its SOM. In this context, capability refers to those services a federate can invoke and/or respond to during a federation execution (e.g., ownership transfer).

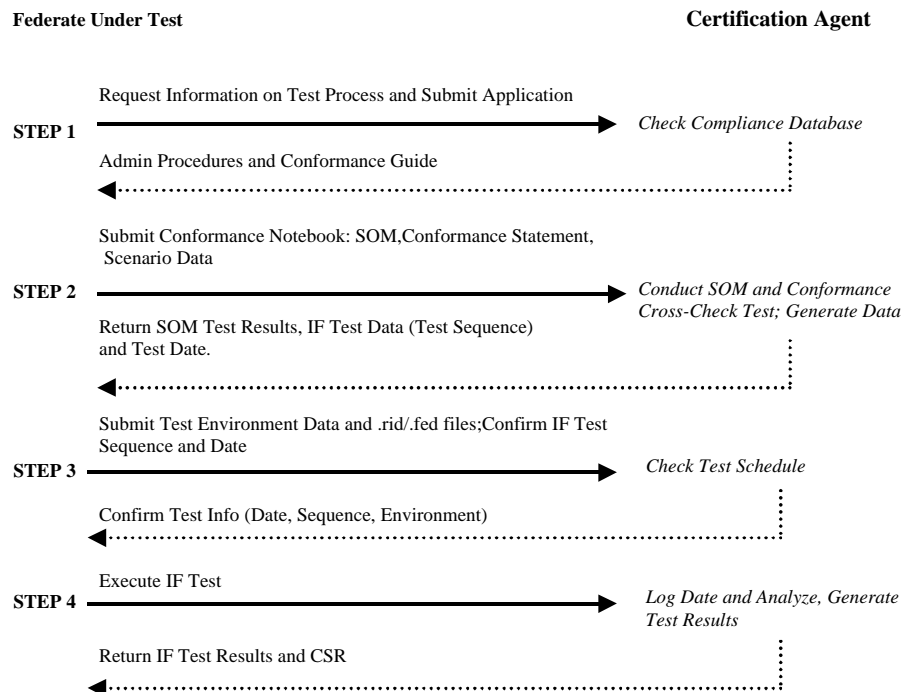


Figure 4.3. HLA Compliance Process

The Interface (IF) Test has two parts: the Nominal Test, which ensures that the FUT can invoke and respond to all services for which it is capable, per its CS; and the Representative SOM (RepSOM) Test, which ensures that the FUT is capable of invoking and responding to services using the range of data contained in its SOM. Figure 4.3 shows the HLA compliance process.

In Step 1, a federate developer requests information on the test process from the official CA by completing an HLA test application on the Web at <<http://hlatest.msosa.dmsomil>>. The CA checks the official Compliance Database to determine the federate's priority for Compliance Testing and, if approved, will respond with a user ID and password for conducting the test. It is important to note that the test process is initiated by the federate developer, not the CA, and it is the responsibility of the federate developer to ensure that the FUT represents a stable, mature release of code. Ideally, the test process should be initiated late in beta testing, so that the actual tests are performed on the release version of the code.

In Step 2, the federate developer submits the Conformance Notebook, which includes the SOM, the simulation Conformance Statement (CS), and optional Scenario Data. The CA checks the SOM for conformance to the OMT (SOM Conformance Test) and, if successful, checks the SOM against the CS for consistency (Conformance Cross-Check). Test results are then returned to the federate developer.

Assuming that the FUT successfully passes the SOM Conformance Test and Conformance Cross-Check, the CA also returns a Test Sequence to the federate developer for IF Testing. The Test Sequence will be based on the Scenario Data submitted with the Conformance Notebook, if available. If the federate developer chooses not to submit Scenario Data with the Conformance Notebook, the CA will arbitrarily create a Test Sequence based on the SOM and CS. The CA will propose a date and time for IF Testing based on other testing commitments in a schedule maintained by the CA.

In Step 3, the federate developer will review the Test Sequence generated by the CA and submit test environment data to the CA. Both the federate developer and the CA will confirm a test date and time.

In Step 4, the IF Test is executed by the federate developer and the CA. The IF Test has two parts: the Nominal Test, which ensures that the FUT can invoke and respond to all services for which it is capable, per its CS; and the Representative SOM (RepSOM) Test, which ensures that the FUT is capable of invoking and responding to services using the range of data contained in its SOM. The CA will log service data from the test, analyze the data, generate results, and return a Certification Summary Report (CSR) to the federate

developer. The CSR is the official record of HLA compliance for the specific version of the federate code tested.

See Section 6.2 for a discussion of the tools used in Conformance Testing.

4.2 FEDERATION TESTING PROCESS

Conformance Testing for the HLA has been defined very specifically in terms of conformance with the HLA Rules, Interface Specification, and Object Model Template. When a federate completes this process for the first time, it can be safely assumed that the federate will still have to undergo federation testing in order to participate in a federation.⁴ This is necessary to ensure that the federate complies with the requirements of the federation. This federation testing will be conducted by an organizational entity specified by the federation sponsor.⁵

Testing a federation of distributed simulations to assure that it functions correctly is a complex process. At the beginning of the FTAM program, there was no generally accepted process to describe the steps needed to test a federation, mainly because few federations at that time had evolved to the testing phase of the Federation Execution and Development Process (FEDEP). Even though more details have been discussed in a paper titled "FEDEP Phase III Examination: Federation Testing, Execution and Feedback," which was presented at the Fall 1997 Simulation Interoperability Workshop (SIW) (Zimmerman & Harkrider, 1997), more guidance is still needed. The Federation Test Process described in this section is a start at providing this more detailed guidance. Section 4.3.1 compares the process presented here with the FEDEP and other existing distributed simulation development processes.

The Federation Testing Process presented here is a multi-phase approach to Federation Testing designed to:

- facilitate discussion of Federation Testing

⁴ Also see (Braudaway & Harkrider, 1997)

⁵ No assumptions are being made in this discussion about the size or goals of the federation. Depending on the goal of the federation, the sponsor could be anyone from the federate developer to the government program manager wanting to use the federate in a large distributed exercise, experiment, or test. The point being emphasized here is that the HLA Compliance Testing Process is often only the first step in getting a federate ready for participation in a federation.

- provide useful information to federation developers to aid federation development planning and execution
- provide a means to compare federation development processes, plans, and experiences.

The process has four phases: Application Testing, Integration Testing, Functional Testing, and Scenario Testing. It was presented at both the Spring and Fall 1997 Simulation Interoperability Workshops (SIW) See (Roberts et al., 1997b).

Figure 4.4 shows the Federation Test Process. The figure also shows the relationship between Federate Conformance Testing conducted by DMSO and the Federation Testing Process that will be described here. Initially, Federates go through Conformance Testing to verify that they adhere to the HLA Rules for Federates. This testing is different from the process a federate will undergo to show that it can be used in a federation, which we label as Federation Testing.

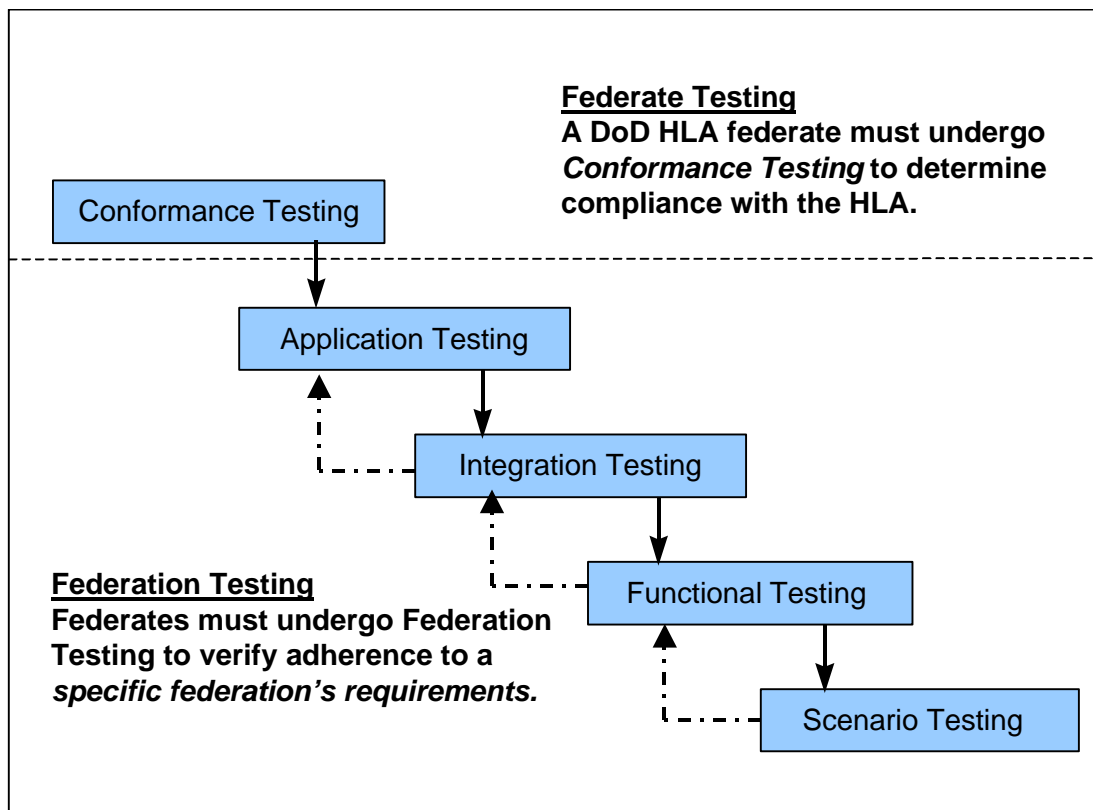


Figure 4.4. Federation Test Process

Each phase of the process is used to describe a kind of testing that needs to be done that assumes and builds on the previous step in the process. But, it is

important to note that this process is not a strictly graduated process; once a federate moves from Application to Integration Testing, it does not mean that no additional Application Tests will be conducted on that federate. In fact, during a later phase, for example in Integration Testing, some Applications Tests may need to be conducted again as more errors are found in the Federate Under Test (FUT).

This discussion of Federation Testing assumes the most rigorous of federation development requirements, where the federation is being developed from a mix of different kinds of federates, modeling at different levels of fidelity. In addition, some federates may already exist and have participated in previous federations, while others are developed specifically to meet the needs of this federation. From prior experience with the development of large SIMNET and DIS exercises, it is well known that the more new software development that is required for the construction of the federation, the more time and effort has to be put into federation testing, VV&A, and operational rehearsal. This is the level of requirement that drives this discussion of federation testing. However, by no means does this mean that this process cannot be used for smaller, simpler, or more persistent federations. (Dahmann, 1998) The federation developer has to weigh the rigor of a formal test process with the specific requirements of the federation and trim down the level of effort required for federation testing to those steps that are useful. This approach to developing a comprehensive process that can be tailored to meet the specific needs of a federation was also used in Lewis' description of the Advanced Distributed Simulation (ADS) VV&A Process. (Lewis, 1997)

4.2.1 Application Testing

4.2.1.1 Definition

The first phase of Federation Testing is Application Testing. In this phase, federates are tested individually to verify that they conform to federation requirements. These requirements should take the form of the following data inputs:

- FOM (Data Representations, Interactions, Interaction Protocols, Timing)
- Federation Agreements
- Fedex Performance Workbook (FPW) Data

These data requirements are discussed in more detail in Section 5.1.

4.2.1.2 Discussion

Application Testing as defined here takes the logical next step past the *syntax* testing performed in the HLA Conformance Tests. In this testing phase, the

semantics of FOM data are tested; federates are tested to see they use the FOM objects and interactions as they were intended. The correctness of data representations, adherence to the order in which events should happen, and the appropriateness of the representation of time are all addressed in this phase of testing.

The definition of the different phases that make up the Federation Testing borrows heavily from system engineering and software management techniques. It makes sense to test individual federates as much as feasible before connecting them to other federates. It also makes sense to limit the number of people waiting for the results of a federate undergoing individual application testing to just the developers and testers involved with that process. This is in contrast with attempts to do individual application testing at the same time as integration testing when other federate developers may be kept waiting for the individual federate fixes. This approach of course should be tempered with the time, effort, and value of accuracy constraints of the federation.

One of the keys to making Application Testing work is to develop test stubs that feed the FUT with appropriate test data so the federate capabilities can be verified. A "test federate" that is designed to be easily modifiable to use the appropriate FOM should be used to help with this process. See Section 6.3 for a discussion of the Federation Test System (FTS), which was designed to meet this need. Conversely, a previously-tested federate could be used as the test stub to support Application Testing, although this approach should be used with caution.

In the case of a federation that will execute using multiple federates at multiple sites, distributed team management is often an issue that has to be managed actively. Often, integration of federates is done remotely with multiple sites being connected to a central control site for long periods. In this case, if feasible, it is wise to do as much Application Testing ahead of time at the developer's site, to not waste valuable network testing time. In order for this to be feasible, Federation Testing procedures should be developed to support individual application testing so that the federation developers can conduct the tests for themselves. If necessary, the application tests can be re-done formally by the integration agent, but if already done beforehand, these tests can be performed very quickly.

As an example, consider the situation where a tank simulation is undergoing Application Testing. Two sample application-level tests that could be performed include:

- Verify that the tank federate can update location object attributes with appropriate data

- Verify that the tank federate can send fire and detonate interactions with correct data

Application Testing addresses individual *federate* semantics, as opposed to *federation* semantics, which are addressed in the next phase, Integration Testing.

4.2.2 Integration Testing

4.2.2.1 Definition

Integration Testing addresses *federation* semantics. The purpose of Integration Testing is to ensure that the following federation data requirements are satisfied *when multiple federates interact*.

- FOM (Data Representations, Interactions, Interaction Protocols, Timing)
- Federation Agreements
- Fedex Performance Workbook (FPW) Data

Integration testing is the first opportunity for the federates to perform complex federation behaviors with each other.

4.2.2.2 Discussion

In the Integration Test phase of the process, it is assumed that individual federates have undergone Application Testing and individually meet federation objectives for the role that they will perform. At this point, the federates are tested together in a progressive manner which we will call 'progressive pair-wise to N-wise' testing, where N is the number of federates that will be used in the federation. Initially, pairs of federates should be tested; then a progressively larger number of federates should be tested together, until in the final stage, all the federates that will participate in the federation will be tested together.

The approach just described is the ideal case. In practice, this approach will not work with very large numbers of federates, so a subset of the possible numbers of test should be conducted, as feasible and practical. Integration Testing should be conducted in an organized fashion by defining a test matrix which summarizes the list of important federate interactions, then using this matrix to systematically test each federate to federate interaction, until all the major interactions are tested. The FOM should provide the data needed to understand the major objects and interactions that can be used to develop the Integration Test Matrix.

In this stage of testing, many of the types of tests conducted in Application Testing will be conducted again, albeit with the federates playing in the roles for which they are required to meet federation objectives. As such, many of the

same test procedures used in Application Testing will also be reused for these tests. In actuality, it is better to think about this test procedure reuse in the other direction: it is often more practical for federation developers to first work on Integration Test procedures before developing Application Test procedures.⁶ This is because it may be easier to break down multiple-federate tests into tests designed to test a single federate's capabilities than to do it the other way around.

As mentioned in the previous section, the management of integration tests for a distributed federation execution at multiple sites is often a difficult process. (Kanewske & Fine, 1998; Roberts, 1995) Accordingly, care has to be taken in the design and timing of the tests. Decisions need to be made if the federate developers all need to be brought together in person to conduct integration, or if the test can be conducted remotely. Sometimes the people and management issues involved in organizing integration testing can be as hard as the technical issues. One of the enablers to successful distributed integration testing could be the use of web-based test procedure management and trouble reporting. The use of the web to support testing is described in Section 5.2.5.

Continuing with the tank federate example introduced in the last section, consider the situation where two tank simulations are undergoing joint Integration Testing. Some sample integration-level tests include:

- Verify that the tank federates receive each other's location object attribute updates
- Verify that the tank federates interpret each other's location object attribute updates (for example, by showing correct visual displays)
- Verify that the tank federates can send and receive fire and detonate interactions
- Verify that the tank federates are advancing time correctly with respect to each other's simulation (i.e., are two simulations in synch as required by the timing requirements of the federation)

⁶ As will be discussed in most detail in the Scenario Testing section, it may be useful to develop Scenario Test procedures first; then work down a level of detail to functional, integration, and application tests. When this approach is used, a natural traceability is maintained and for example, the relationship between a specific application test and a scenario test will be clear.

Integration Testing also needs to address the representation of the synthetic environment as implemented by the federates. During this phase, correlation of terrain and other synthetic environment databases need to be tested.

At the end of integration testing, all of the federates required for the federation execution should be able to communicate meaningful information using the objects and interactions of the FOM. At this point, the individual federates and the federation as a whole can be tested with more rigor to verify whether they meet the functional requirements of the federation. This next phase of the Federation Testing Process is Functional Testing.

4.2.3 Functional Testing

4.2.3.1 Definition

In the Functional Testing phase of Federation Testing, individual federates and the federation as a whole are tested to see that they can fulfill their intended purpose. Functional Testing can only be performed *after* Integration Testing is complete because it is only after all the federates can communicate using the same FOM in the same synthetic environment that you can test their ability to perform the function for which they are required.⁷ Using the same logic, this is realistically the first stage of the testing process where the whole federation can be tested to see if it will meet the requirements of the federation sponsor. It is at this stage of the process that appropriate federation fidelity, accuracy, and resolution issues are verified. In addition, at this stage, the detailed implementation of algorithms could be tested. The main HLA data input for this stage is:

- Federation Requirements

⁷ Functional Testing is related, but more complex than the term 'Unit Testing' used in software engineering. Unit Testing in this context describes the steps that the federate developer takes during the development of the federate to verify that it meets its stated design requirements. See Section 4.2.5 for more on Unit Testing. Although this Unit Testing is valuable, and even required, it still does not go to the extent needed for federation Functional Tests.

In addition, there is another kind of functional testing that should occur between Unit Testing and Functional Testing; labeled by Roberts in "Integration Test of Distributed Interactive Simulation Exercises" as the *pre-integration functional test*. (Roberts, 1995) During this test, which occurs before federation integration, the user of the system verifies that the FUT can perform the function required in the federation by watching the federate in stand-alone mode performing the function. For example, if the requirement is for the tank simulator to have voice communications capability, this functionality should be verified before integrating the federate. As has been demonstrated by experience in War Breaker DIS exercises, if "...deficiencies in the basic functionality of a simulator are found this early in the process, then there is possibly enough time to fix them before the scheduled exercise." (Roberts, 1995)

These federation requirements could be formal or informal, but must be used to drive Functional Testing of the federation.

4.2.3.2 Discussion

Functional Testing is the third step of Federation Testing. In this phase of testing, it is assumed that the federates under test (FUTs) have successfully completed HLA Conformance, Application, and Integration Testing. This assumption is required because of the underlying concepts of distributed simulation. In a distributed simulation system, each component of the system only simulates a part of the real world. The simulation components *rely* on each other to form a complete virtual world in which they can interact. Based on this reasoning, it is only after all the federates required for the federation are integrated into the same virtual environment that functional verification of the federation can occur.

Functional Testing is the first phase of the process where issues such as training requirements, usability, and fair fight are tested. Although these issues will be addressed in the federation and individual federate design processes, this is the first phase of the Federation Testing process where they can be *tested*.

Extending the example started in the previous sections, it is at this stage of the process where the tank federate would undergo testing to verify that it can play the role of a tank to a level of fidelity and appropriateness to meet the needs of the federation. Sample tests could include:

- Can the operator of the tank simulator or the ModSAF tank simulation use its infrared sights in a night-time scenario to see enemy targets on the battlefield at the correct distance, accounting for the overcast skies and the phase of the moon?
- Can the tank simulation/simulator fire a specific munition at the correct sustained rate and be restricted by realistic ammunition logistics re-supply limitations?
- Can the tank semi-automated forces simulation maneuver correctly around obstacles, taking into account tactically correct maneuvers?

Obviously, from the examples, it can be seen that the procedures that would be used in a simulator would be different than would be used for semi-automated forces simulation (ModSAF), and both of these would be different from what be used for a live, instrumented tank. For example, in ModSAF, human behaviors are encoded in the simulation so these behaviors have to be tested to see if they meet the tactical correctness requirements of the federation. In a simulator, the operator provides this tactical behavior while the tank simulator provides a virtual representation of the vehicle and weapon systems behavior. As another example, for a ModSAF simulation, the operator does not have to tell the tank to fire on the opposing tanks. The operator only has to specify that a tank has free

fire permission, and the tank will decide on its own whether to fire on another. Additionally, if the tank is told to travel to a point on the other side of a river, it will automatically change its course to travel across a bridge that isn't on its original route. The key at this stage of Federation Testing is to test based on:

- the requirements of the federation
- the role the federates have to play in the federation
- and the capabilities of each individual federate to perform the function.

4.2.4 Scenario Testing

4.2.4.1 Definition

The final step in the Federation Testing Process is Scenario Testing. It is at this phase of the process where the scenario for the federation exercise, test, or experiment is used to verify that the federation as assembled will be able to meet its original goals. This phase can also be dubbed *operational testing*, where the federation is used to practice the scenario called for in the federation design. As mentioned, the main HLA data input for Scenario Testing is the:

- Federation Scenario

The scenario will drive the specific test procedures used in this phase.

4.2.4.2 Discussion

Scenario Testing is more focused than Functional Testing, because only the scenario for the specified federation execution will be used to drive testing. This is as opposed to Functional Testing where the basic capabilities of federates are tested according to the federate's role in the federation.

Using these definitions, if all prior phases of Federation Testing have been completed and no changes have been made in the federates, then multiple scenarios could be run in multiple federation executions and only Scenario Testing would be required to support these efforts. This is the ideal case of reuse and is the idea that supports the need and use of persistent federations that are maintained to meet an ongoing federation need.

It is at this phase of the Federation Testing process where federates will attempt to perform as they would in a real federation execution. In this dress rehearsal, all the kinks are worked out of the federation as well as the scenario. There may be times when the scenario should be changed to reflect the capabilities of the federates being used, as long as the federation requirements are still being met.

Continuing the tank federate example started in previous sections and assume a simple scenario as follows:

There are two tanks in a training scenario. Tank A is to travel to a point approximately one mile away from the assigned objective X. Behind hill Y along the route to objective X and hidden from view lies tank B. Initially, neither tank can detect the other. At some point, the tanks will detect each other, and engage or retreat. Assume the objective of the scenario is weapon system training for the operators in the simulator which is playing Tank A.

In this simple case, Scenario Testing will verify that the federates involved can perform the roles assigned to them in the scenario. For example, if tank B is being played by a ModSAF system, the test should verify that the tactics parameters loaded in the simulation meet the needs of the training being conducted. If Functional Testing was done well, we could assume that the ModSAF simulation can perform in the prescribed role, but its ability to perform in this specific scenario using the right parameter input files will be tested in Scenario Testing.

In the above example, there are only two tanks involved in the scenario. Testing whether these two tanks passed the Scenario Test is straightforward. However, what if there were a large number of platforms in the scenario? For instance, what if the test involved a platoon of tanks taking a hill? How should the test be conducted to show that the federation could meet the requirements specified in the scenario? For this case, there could be an agreed-upon statistical threshold for the platoon of tanks to pass the test. For example, for a federation simulating a platoon of tanks taking a hill, the Scenario Test would be successful if x% of the tanks reached the top of the hill. Test procedures for scenario testing need to support these types of scenario acceptance thresholds.

One conclusion that should be immediately obvious from this discussion is the need for automated tools to support this process.⁸ Even though subject matter experts can view simulation behavior using representative visual displays, it would be difficult to perform any of these tests systematically with a large number of federates. Tools such as the analysis federate component of the Federation Test System (FTS) would be useful to meet this testing automation requirement.

As has been shown in this discussion, several data inputs are needed to support the automaton of this type of testing. These include:

⁸ Additional support for this assertion is shown by the experience gained from early federation prototypes. (Graffagnini, 1997).

- Scenario descriptions in a computer-readable formats
- Scenario requirements that define acceptance thresholds for scenario behavior
- Scenario test procedures in computer readable formats that can use the scenario descriptions and acceptance thresholds

As was discussed in an earlier section, it may be useful to develop Scenario Test procedures first; then work down a level of detail to Functional, Integration, and Application tests. When this approach is used, a natural traceability is maintained and for example, the relationship between a specific application test and a scenario test will be clear. (For more on data requirements and formats as well as method requirements for Federation Testing, see Section 5.1.)

4.2.5 Other Types of Testing Within the Federation Testing Process

The Federation Testing Process as described contains several testing phases, namely Application Testing, Integration Testing, Functional Testing, and Scenario Testing. Each of the sections describing these test phases discussed the types of tests that are conducted. However, some types of testing that are common to software and system engineering may have been omitted from the discussion. This section is designed to address that deficiency in the Federation Testing Process description. The following paragraphs define common 'types' of testing and describe where they would be used in the overall Federation Testing Process.

4.2.5.1 Unit Testing

Unit Testing describes the steps that the federate developer takes during the development of the federate to verify that it meets the need for which it was originally intended. (Clay, Roberts, & Stueve, 1995) In unit testing, the federate developer will test the federate using stub code to represent its interaction with other federates. It is likely that the developer will also test with a duplicate copy of the federate as well as any other federate that is available to help with the sanity check. Unit testing should be conducted by the federate development organization before submitting the federate for Federation Testing. Unit testing should also be conducted when a change is made in the federate during Federation Testing to meet the needs of the federation. It is important to distinguish between unit testing, which is conducted by the federate development organization, and Application Testing that is a part of federation's Federation Test process. As discussed earlier, Application Test procedures could and should be provided to the federate developers ahead of time so that they can be performed independently. However, this still does not alleviate the requirement of the federation to verify the basic data exchange and comprehension requirements that are tested in Application Testing.

4.2.5.2 Exception Testing

Exception Testing is a method of testing where the approach is to test all the things that the system should not do. For example, with DIS testing, there is a list of common mistakes that are made in implementing the standard. The process of verifying that the simulation handles these common mistakes correctly would be called exception testing. Exception testing as described here is a useful test method that could and should be used throughout the Federation Testing Process.

4.2.5.3 Performance Testing

Performance Testing in the context of Federation Testing is the verification that the federation, individual federates, and the supporting infrastructure (workstations, RTI, and communications network) meet the performance requirements of the federation. Performance testing is applicable to all phases of the Federation Testing process. The testing of network, RTI, and application latencies can all be considered performance testing. See the "High Level Architecture (HLA) Performance Framework" paper in the Spring 97 SIW Proceedings for more information on performance measurement and testing approaches for the HLA. (Dahmann et al., 1997) Also, see (McKee, 1998; White, 1998; Wuerfel, 1998)

Another related testing term is *Stress Testing*, which is performance testing conducted by increasingly loading parts of the system to see if and when it breaks. For example, CPU and LAN/WAN loading tests can be called stress tests.

4.2.6 Automation of the End to End Federation Testing Process

One idea that has been suggested by several preceding sections is the need for automation to support Federation Testing. One of the main benefits of automating Federation Testing would be the ability to reduce the number of separate testing sessions by combining different types of tests into the same test session. The ultimate goal would be to run the FUTs simply through a scenario-level test and have the test system automatically verify the required Conformance, Application, Integration, and Functional Tests. The key to realizing this goal is understanding the relationships between the phases of testing and the data requirements of each process phase. Unfortunately, these relationships are not straightforward and vary depending on the federation.

In Scenario Testing, the data being tested is scenario data, which could consist of the following:

- high level objectives (example: mission rehearsal operations orders, training objectives, test objectives, experiment hypotheses)
- initial conditions of the objects
- time-phased (objective, entity, state) -tuples

The objective of Scenario Testing in this instance would be to verify that each part of the time-phased objectives are potentially achievable by the simulations being tested, based on the high level objectives and the initial conditions.

With the above scenario definition, Functional Testing would verify that a simulation can perform the required role specified in the scenario. For example, in a training scenario where the goal is to practice naval aviation operations under stress, the sample objective could be: Fixed Wing Aircraft A, take off from aircraft carrier 1, flying at night, rendezvous with tanker H, refuel in flight, avoid surface-to-air-missile (SAMs), attack target 18, return to base. In this case, the data that could need to be verified in Functional Testing include: verification that the aircraft simulator can perform takeoffs, function in night time operations, refuel, and avoid SAMs effectively enough for the satisfaction of the user of the training exercises.

Sample integration data that could be verified in relation to the above Functional Tests include: verification that the plane federate and aircraft carrier federate can exchange position updates and collision interactions as necessary so that the takeoff interaction can be represented to the satisfaction of the interoperating simulations.

In Application Testing, the following data could be tested to support the integration described above: verification that the plane federate can update its position attributes at an appropriate rate and send collision interactions at the appropriate time.

In HLA Conformance Testing, appropriate testing data in support of the Integration Testing described above would include verification that the plane federate can perform the Publish Object and Update Attribute services correctly.

As shown, the data tested in the above example are very much correlated. In fact, with careful design of the testing process, each higher level test could be made up of atomic tests of the underlying tests. The question that must be addressed is: How can we take advantage of the hierarchical nature of the above testing phases so that some steps of the process can be automated, thereby reducing the time and resources needed to perform testing?

4.3 RELATIONSHIP BETWEEN EXISTING PROCESSES AND THE FEDERATION TESTING PROCESS

To stimulate discussion on Federation Testing, it was important to compare the above model to others currently being used. The goal is for this comparison to help in refining the Federation Testing process for future efforts, as well as point out important components that may be missing from the process as proposed. The following sections describe the relationship between the Federation Testing process and the FEDEP; between the Federation Testing and VV&A processes; and Simulation Testing and Test and Evaluation.

4.3.1 Relationship between Federation Testing and the HLA FEDEP

In the current version of the FEDEP, little is said about the implementation and testing phases. These phases were discussed in more detail in a paper titled "FEDEP Phase III Examination: Federation Testing, Execution and Feedback," which was presented at the Fall 1997 Simulation Interoperability Workshop (SIW) (Zimmerman & Harkrider, 1997). A representation of the relationship between Federation Testing and the FEDEP is shown in Figure 4.5.

In this paper, Federation Testing is described as a function of three parts: Compliance Testing, Integration Testing, and Federation Testing. FEDEP Compliance Testing refers to the process of testing the application for the correct implementation of HLA requirements. Likewise, Compliance Testing in the Federation Test process consists of testing for conformance to the IFSpec and OMT. Thus, there is a straightforward comparison between FEDEP Compliance Testing as defined in the paper referenced above and Compliance Testing as part of the Federation Test process we are defining. It is important to note that in FEDEP Compliance Testing, the syntactic elements of the FOM are tested and not the semantic elements.

FEDEP Integration Testing refers to "...bringing all of the pieces of the federation together and assessing their ability to interoperate." (Zimmerman & Harkrider, 1997). This step can be compared directly to the Integration Testing step of the Federation Test process. The FEDEP testing process does not specifically call out the need for application testing as defined in the federation testing process. Before Integration Testing can occur, it is important to test the semantic aspects of the FOM for individual federates so problems can be isolated and corrected. It would be more difficult to do this in a 'big bang' testing integration testing approach.

The final testing step in the FEDEP is called Federation Testing. This final interoperability test assesses a federation's ability to meet the user's objective for the test. The FEDEP paper describes this step as made up of four substeps: objective, scenario, VV&A requirements, and security requirements. (See Figure 4.5) Objective testing is related directly to the Functional Testing step of the

Federation Test Process. Likewise, Scenario Testing is related directly to the Scenario Testing step. As opposed to what is explained in the paper, both VV&A and security should not be limited to the Federation Testing phase of the FEDEP. VV&A and security need to be involved throughout the entire testing process (as explained in the following section), as well as throughout the FEDEP process.

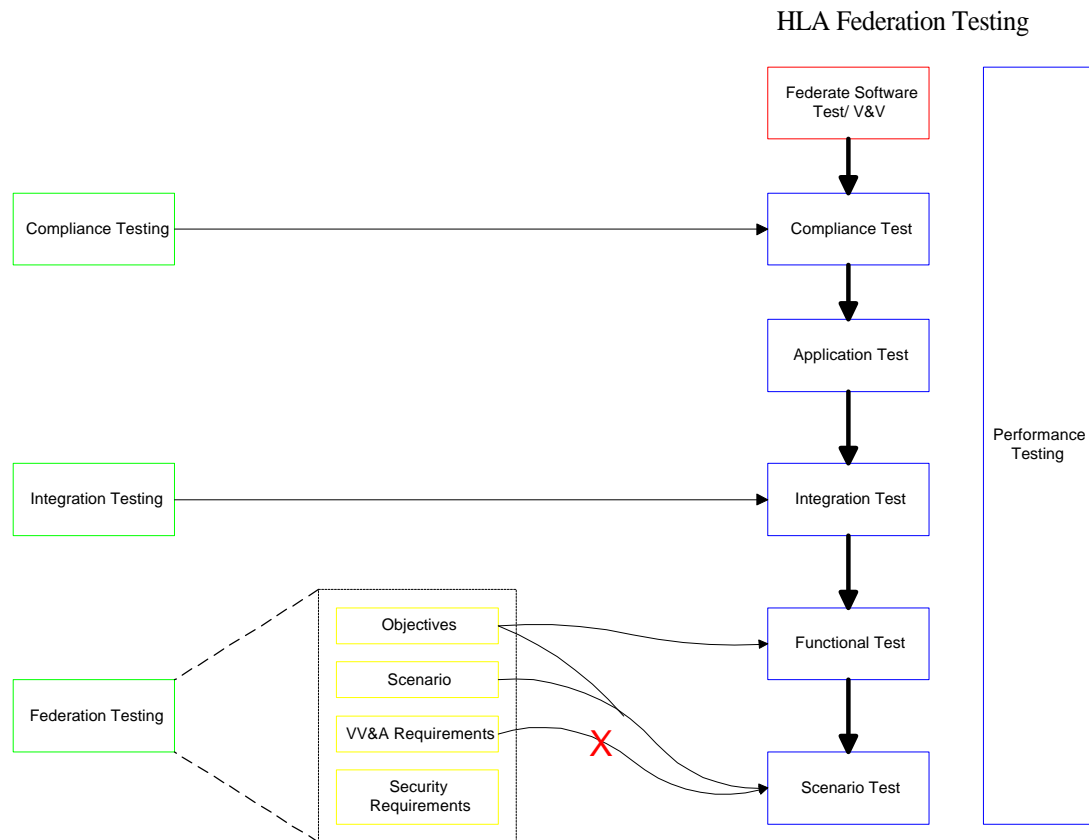


Figure 4.5. Federation Testing Process and FEDEP Comparison

Thus, a comparison between the Federation Testing Process presented in this report and the current FEDEP testing process shows that the former covers all aspects of the latter. In fact, there are certain components that the current FEDEP leaves out, such as the need for one-on-one semantic testing before Integration Testing. Also, as alluded to previously, the current inputs into Federation Testing are not enough to perform complete testing. (Certain inputs, such as OIPs, are explained in later sections.) Further refinement of the FEDEP is needed to cover the details necessary for effective Federate and Federation Testing.

4.3.2 Relationship between Federation Testing & VV&A Processes

Discussions in the VV&A and Test Forums during the Spring 97 SIW reinforced the need for a process that describes Federation Testing steps. This discussion raised questions about how testing steps fit into the VV&A. This section is designed to address these questions.

Several resources exist for understanding VV&A in the context of HLA. The IEEE Recommended Practice for Distributed Interactive Simulation--Verification, Validation, and Accreditation (IEEE, 1997b) establishes the recommended VV&A approach for participants in a DIS exercise and provides "how to" procedures for planning and conducting DIS exercise VV&A. A DoD (VV&A) Recommended Practices Guide also has been published. It provides background and information on principles, processes, and techniques recommended for use in DoD VV&A efforts that support program initiatives in the analysis, acquisition, and training communities. This guide can be found on the Web at <<http://www.dmsi.mil/docslib/mspolicy/vva/rpg/>> (DMSI, 1996a). It supports the implementation of DoD Instruction 5000.61, titled "DoD Modeling and Simulation (M&S) Verification, Validation, and Accreditation (VV&A)," April 29, 1996, which can be found on the Web at <<http://www.dmsi.mil/docslib/mspolicy/vva/dodifin.doc>> (DoD, 1996).

An appropriate way of showing the relationships between Federation Testing and VV&A is to compare the processes. We do this by comparing the Federation Testing Process described in this report and the IEEE VV&A process for advanced distributed simulation, (ADS), summarized and discussed in (Lewis, 1997).

VV&A is a process that should span the entire life cycle of a federation. In the ADS development model, VV&A is made up of two phases: the Planning and Requirements phase and the Design and Development phase. Subsequently, the Design and Development phase is made up of several stages: Conceptual Model, Preliminary Design, Detailed Design, Construction and Assembly, and Integration and Test. Federation Testing only relates to the Construction and Assembly process and the Integration and Test process. All of the other stages fall under other phases of the FEDEP model (i.e. Conceptual Analysis and Federation Design).

Thus, the focus of this section is to depict the relationship between Federation Testing and the VV&A steps involved in the Construction and Assembly and the Integration and Test stages of the ADS development model. A visual representation of this relationship is shown in Figure 4.6.

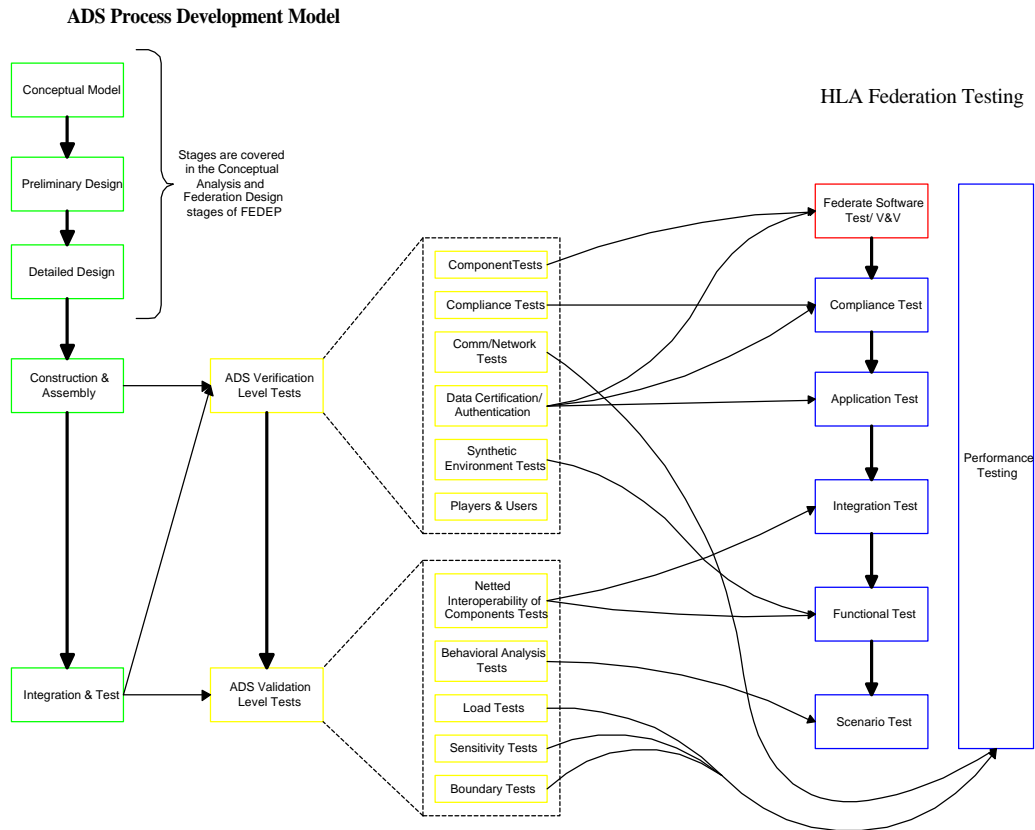


Figure 4.6. Federation Testing and VV&A/ADS Process Comparison

The Construction and Assembly stage is the process of putting the federation together to bring it up to working order. The Lewis paper outlines six verification levels involved in this stage: Component Test, Compliance Test, Communication/Network Test, Data Certification/Authentication, Synthetic Environment Tests, and Players and Users. Since the Component Test assesses the accuracy of the simulation code, there is a direct relationship to the what is commonly known as software Unit Tests. These tests are conducted by the federate developer before joining a federation, and so are in general outside the scope of Federation Testing as described in this report. This tests have been added the Federation Testing Process diagram above for convenience of this discussion and have been labeled Federate Software Tests.

Compliance Testing involves looking at the data formats for conformance to the specifications. Thus, this test has a direct relationship to the Compliance testing phase of Federation Testing (even though if data content is verified, these tests should be termed application testing.) The Communication/Network Test ensures that the network is working properly. This test is covered under Performance Testing in the Federation Testing Process, which involves CPU loading, LAN/WAN loading, latencies, and time synchronization. Since Data Certification/Authentication tests the correctness of data as it is passed between

federates, this process covers several areas of Federation Testing. The Federate Software stage verifies that a simulation is making the correct calculations. The Compliance Test verifies that the data is sent in the right format (i.e. HLA IFSpecs), and the Application Test verifies that the data sent is meaningful to the federation (i.e. semantic). The Synthetic Environment Tests assess whether the different federates can share a common environment. These tests typically involve two federates. When two or more federates are involved, there is a direct link between these tests and Functional Testing, which tests for accurate environment representation, among other things.

The Integration Test phase of the Federation Testing Process verifies the functional and performance aspects of federates within a federation and validates the interactions, behaviors, and effectiveness of the federates within an exercise or scenario. Thus, when tests are run on only portions of the federation, this phase falls under the verification level. When the entire federation is tested, it falls under validation. The first step of validation is the Netted Interoperability of Components Test, which ensures that the entire federation is communicating and interacting correctly. This test is comparable to the Integration and Functional Tests. The Behavioral Analysis Tests simply assess the behavior of the model and whether it depicts the real world to the level needed. These tests are linked directly to the Scenario Testing process of Federation Testing. The final three stages of validation are Load Tests, Sensitivity Tests, and Boundary Tests, which are linked directly to Performance Testing.

Thus, there exists an important relationship between VV&A and Federation Testing. The framework that was developed to verify, validate and accredit Advanced Distributed Simulations (ADS) and HLA Federation Testing relate in that all aspects VV&A discussed seem to be covered at some stage in the testing lifecycle. Further refinement of both the VV&A process and the Federation Testing process is needed to make it simpler for the user to understand more clearly how the processes are related.

4.3.3 Application of the Federation Testing Process to Federations

Another important comparison that needs to be made is between the Federation Testing Process and existing federation testing processes. This has (effectively) already been done for the HLA protofederations in Section 4.3.1 because the FEDEP process was developed from the lessons learned by the HLA protofederations.

Due to the relative immaturity of existing federations, not much formal federation testing has been done.⁹ Because of this, we compare the Federation Testing Process with the Joint Training Confederation (JTC), a mature set of distributed simulations using the Aggregate Level Simulation Protocol (ALSP), which is scheduled for HLA migration.

The ALSP JTC testing process is described in "The ALSP Joint Training Confederation: A Case Study of Federation Testing", a paper presented at the Fall 1997 SIW. (Page & Babineau, 1997) Figure 4.7 is a visual representation of the relationship between the ALSP JTC Federation Testing process and HLA Federation Testing.

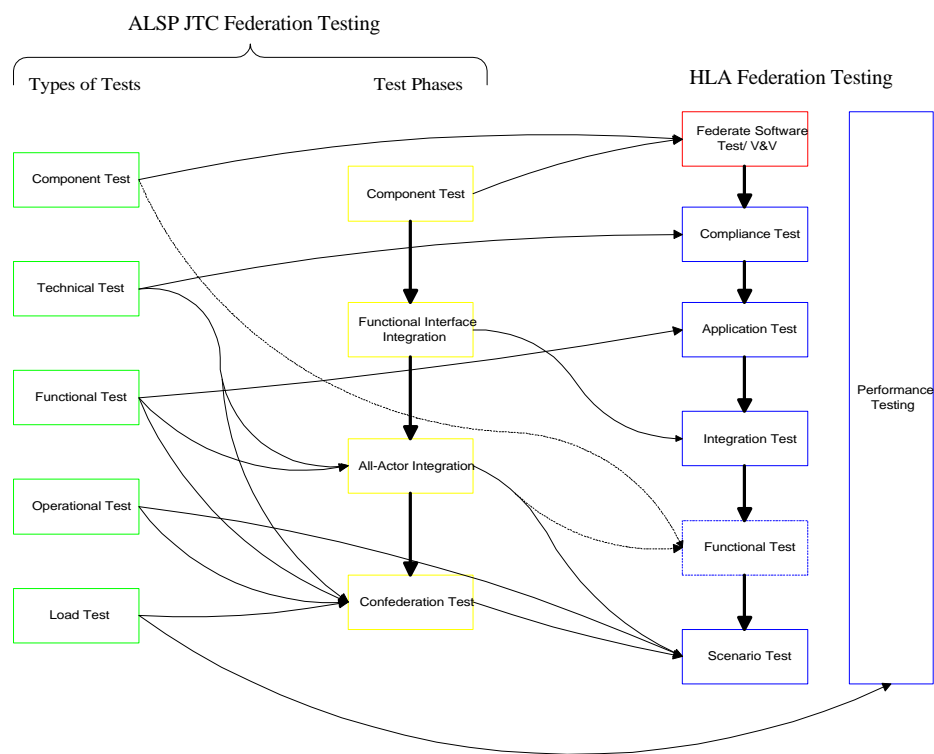


Figure 4.7. ALSP JTC Testing and the Federation Testing Process

The ALSP JTC Federation Testing process is made up of four phases: Component Test, Functional Interface Integration (FII), All-Actor Integration (AAI), and Confederation Test. Analogous to these phases are five different test plans: Component, Technical, Functional, Operational, and Load Tests.

⁹ With the exception of STOW 97 (Kanewske & Fine, 1998) and the HLA C2 Experiment. (AEgis, 1997).

As shown in the figure, many relationships between the Federation Testing process and the test phases and/or test plans of the ALSP JTC can be observed. We first discuss the 'Types of Tests' column in the figure above. This set of items is really the list of *Test Plans* developed for the confederation. Specifically, the JTC Component Test Plan refers to individual Verification and Validation (V&V) that must take place on the simulations. These tests can be compared directly to the unit testing described in Section 4.2.5. Next, the Technical Test Plan deals with "matters of conformance and compliance with the ALSP protocols," (Page & Babineau, 1997) which is related directly to the Compliance Test. The JTC Functional Test Plan is designed to verify that the specification for the confederation has been correctly implemented. This relates directly with Application Testing in the Federation Testing Process. The JTC Operational Test Plan evaluates the behavior of the JTC against a set of predefined objectives and has a direct link to Scenario Testing. Finally, the Load Testing Plan, which evaluates the performance of the FTC, is related to Performance Testing in the Federation Testing process. Some important missing links in this discussion are the fact that there is no obvious link between the test plans and Integration and Functional Tests in the Federation Test Process. This may be confusing, but is not a real issue, because these tests are covered in the actual test events that are described in the following paragraph.

As indicated in the Paige and Babineau paper, the JTC process is integrated and tested in four stages. Component Testing, as explained above, can be described as federate Unit Testing or as indicated in the figure above, Federate Software Testing/V&V. The Functional Interface Integration (FII) stage tests how subsets of the JTC federation communicate, which is directly related to Integration Testing. The All Actor Integration (AAI) Test focuses on whether the federation as a whole meets certain specifications, thus falling under Functional and Scenario Testing. Finally, Confederation Testing involves assessing the realism of the federation as it operates and has a direct relation to Scenario Testing.

5. DATA AND METHOD REQUIREMENTS FOR FEDERATION TESTING

Section 4.2 discussed a federation testing process which is proposed to guide the federation developer in testing whether the federation meets the needs of the federation user. Section 5, Data and Method Requirements for Federation Testing, discusses the data and method enablers which will need to be in place in order for an automated federation testing process to be successful.

The section on data requirements reiterates the data requirements discussed in the Federation Test Process. Specifically, this section discusses the need for FOM data, for Federation Agreements, for FPW data, and for Object Interaction Protocols. Each of these data types is described and examples are given of each.

The section on method requirements discusses the need for standard interchange formats, the need for a test procedure interchange format, as well as the need for using the MOM and the internet in facilitating testing.

All of these data and method issues need to be addressed in order for federation testing to be as seamless, integrated, and automated as possible.

5.1 DATA REQUIREMENTS FOR FEDERATION TESTING

This section discusses several HLA data requirements for federation testing. These are Federation Object Models (FOMs), Federation Agreements, the Fedex Planning Workbook (FPW), and Object Interaction Protocols. This section will provide detail on what the requirements are and why they are important to Federation Testing.

5.1.1 Federation Object Models (FOMs)

The Federation Object Model is the main data input to Federation Testing. Whether it is the use of SOMs in Application Testing, the use of a FOM made up of multiple SOMs for integration testing, or the use of the final Federation Object Mode for Scenario Testing, the FOM in the OMT format is a key enabler for automated Federation Testing. Most of the development effort spent in Federation Testing to date -- more specifically Application Testing -- has been in using a FOM to design test procedures. The idea is that similar FOMs will be able to use similar test procedures. Thus, in the future, it is possible that a federation's object model could be a benchmark for utilizing the federation's set of test procedures.

The following sections discuss the DMSO-sponsored Object Model Library (OML) and Object Model Data Dictionary (OMDD), both initiatives which will help the federation developer and tested.

5.1.1.1 Central FOM Library

The Model and Simulation Resource Repository (MSRR) referenced in the FEDEP is a distributed network of resources, organized by resource categories and maintained by the resource owners. The MSRR catalog contains descriptions about various types of resources related to modeling and simulations. These types of resources include models, simulations, tools/utilities, documents, and data sources.

The MSRR is accessible through the Web at <<http://www.msrr.dmsomil.com>>. It includes the Object Model Library (OML), located at <<http://www.omlibrary.epgc4i.com>>. The OML was opened to users in late October 1997 and is now in release build 7.1 (12 November 1997), supporting OMT DIF 1.1. Users can search and browse its SOMs and FOMs, and anyone can check out OM from the library using the download mechanisms available on the Web site. Users also can upload an OM to the library, although they must register with the OML first to ensure accountability. Uploading can be accomplished using a Web browser or anonymous FTP to the OML Upload Area. Once the OM (in OMT DIF 1.1 format) has been uploaded to the OML, it can only be checked into the library by a registered user. The OML performs a simple parsing check to ensure that the OM is in the proper DIF, but performs no other validation of the model.

At the time of this writing, the OML included the following:

- CCTT SAF SOM
- ModSAF FCS SOM
- CTAPS SOM
- Eagle SOM
- NASM AP SOM
- Real-time Platform Reference FOM
- Engineering Federation FOM
- Joint Training Confederation FOM
- NASM/AP SOM
- Joint Training Federation Protodefederation FOM

Additional FOMs expected to be in the OML include:

- F-14D FOM (NAWC-TSD)
- Countermine Component FOM (Army / NVESD)

The OMDD was opened to AMG members for alpha testing at AMG22 (10-11 December 1997). Capabilities of the OMDD include browse, search, manage user selections of OMDD components persistently, and export selected OMDD components in OMDD DIF format.

OMDD elements include:

- Classes
 - names, associated terms, definitions, notes, and status
- Generic elements (attributes and parameters)
 - names, associated terms, definitions, notes, and status
 - data type, units of measure (multiple representations)
- Complex data types
 - names, fields, associated generic elements, and status
- Enumerated data types
 - names, enumerators, representations, notes, associated terms, and status
- Interactions
 - names, associated terms, notes, and status

Official release of the OMDD is expected in Spring 1998, contingent upon the results of the OMDD experiment and early user feedback. The OMDD experiment is a trial use of the OMDD and the updated FEDEP 1.1 to create a federation. Three approaches are being considered: bottom-up, single SOM/merge SOMs, and reference FOM. The bottom-up approach ensures consistent data definition and offers multiple options in the structuring of the federation, while relying heavily on the OMDD. The single SOM/merge SOM approach is closest to existing federate implementation but can lead to inconsistencies in naming. In the reference FOM approach, it is generally easier to remove rather than add elements to the federation. In addition, it is likely that different federation developers implementing the same types of objects would develop compatible FOMs, so the reference FOM should be "comfortable" to other developers in the same application area. The drawback to this approach is that its generality may leave an unnaturally deep hierarchy that serves no purpose in a particular federation. The initial conclusion of the OMDD

experiment was that the existing FEDEP and tools support all three approaches, indicating flexibility and utility.

5.1.1.2 RPR FOM

The DIS family of standards (identified as of the IEEE 1278 series of standards) defines a set of mechanisms that deliver a high degree of interoperability among a specific class of simulations (IEEE, 1993). These simulations are characterized as "real-time" in that they use a time standard directly traceable to Coordinated Universal Time (UTC). They are characterized as "platform" because they share a conceptual level of representation, called "entity," which is aligned with the military concept of a weapon system's platform. Many simulations have been developed or modified to use DIS mechanisms, and they are used in several M&S domains. They can be assembled into federations without any modification, a capability that has been demonstrated for federations in excess of 50 different models.

The problem facing DIS standard simulations is achieving the same level of interoperability using the HLA. To achieve the needed level of interoperability, a standard that addresses the content of the FOM for such a federation is needed. Reference FOMs were developed to serve that part of the M&S community. A reference FOM fully defines the representations and interactions of its federations. As such, federates can be individually tested to show compatibility with federations using the reference FOM. Then, they can be joined into a specific federation to accomplish a specific simulation run without any representation or interaction disconnects to cause software changes. The FOM that accomplishes a complete mapping of DIS is called the Real-Time Platform Reference FOM (RPR FOM), and it makes an ideal starting point for federations whose requirements are *nearly* met by DIS. They can make a relatively small change, then correspondingly small changes in federates from the RPR FOM part of the MSRR. The RPR FOM may not be the only reference FOM, and other M&S communities with common implementation techniques are encouraged to build their own reference FOMs, as opposed to modifying this reference FOM.

5.1.1.3 Relevance to Federation Testing

The FOM is a critical component of Federation Testing, since it is the key document through which a federation is described. Most of the development effort spent in Federation Testing -- more specifically Application Testing -- has been in using a FOM to design test procedures. The idea is that similar FOMs will be able to use similar test procedures. Thus, in the future, it is possible that a federation's object model could be a benchmark for utilizing the federation's set of test procedures.

5.1.2 Object Interaction Protocols

An Object Interaction Protocol (OIP) is a set of ordered atomic events that define a behavior. These behaviors can be defined in terms of initial conditions, a transition, and a set of final conditions. For HLA, OIPs can be seen as a means to describe a desired sequence of service invocations using a specified set of objects and interactions.

5.1.2.1 Background

In the early development of the FEDEP, the idea of using OIPs arose as an additional step in the definition of federation data. According to descriptions of the FEDEP, originally documented on the DMSO Web site before the FEDEP document was written (DMSO, 1996c), "...the Protocol Catalog is envisioned as an on-line database that will contain standard definitions and formats of data exchanged between distributed simulations. This database will contain definitions and formats at all levels from atomic elements to Object Interaction Protocols (OIP) to families of such protocols." (DMSO, 1996b) However, this concept never came to fruition in the 1996 protofederation experiments or in subsequent versions of DMSO documents because of a lack of commitment by the user community to OIPs.

5.1.2.2 OIPs for Federation Testing

In the requirements phase of the STRICOM FTS IDT (see Section 3.3.1), OIPs were proposed as a required input into the Federation Testing process. Although it was too early in the maturation of HLA and the IDT for OIPs to become a priority, the concept of using OIPs is still an important one because, for the testing process to be successful, specifications of required behavior are still needed.

It is important for the user community to establish a need for OIPs, to define if and how they fit into the OMT, and to determine the feasibility of defining a library of reusable OIPs. *The extra work required to develop, implement, and test the OIPs is important when there are interactions where order is important, and especially when there are critical interactions that influence the outcome of the exercise.*

An important step in supporting the increased use of OIPs is identifying an acceptable interchange format and adopting a set of conventions to be used with the FOM. Adoption of a standard will offer several benefits:

- Clarity: A standard format for OIPs would make the specification of interaction sequences in HLA documentation more clear.

- Interoperability and Reuse: OIPs expressed in the standard format could be reused across similar FOMs.
- Testability: A standard format for OIPs would allow them to be tested via automated tools, reducing the time and money required for federation implementation.
- Code Generation: OIPs could possibly be used to automatically generate required behavior sequencing in federates.

Once a file format has been established, federate developers can construct the required behaviors in terms of OIPs. Test procedures then can be defined in terms of identifiable data (FOMs) and behavior (OIPs).

5.1.2.3 Describing and Documenting OIPs

One way in which OIPs can be represented is through Message Sequence Charts (MSC). An MSC is useful in showing an ordering of events and the state of an object at a certain time. Also, MSCs can be defined on different levels. For example, during Application and Integration Testing, the interest is in the HLA service invocations. OIPs can be used to describe the service calls between federates for any scenario. Figure 5.1 shows an OIP depicting a fire and detonate series between two federates.

It should be noted that the dashed lines in the diagram signify that these events can occur in any order. The diagram also has an associated standard textual representation, which will prove useful when trying to automate the test procedure development process. An abridged version of this representation is shown in Section 9.2.

The OIP also can be used to describe higher level behavior that would be useful in Functional and Scenario Testing. Figure 5.2 shows an MSC in which an infantry company falls under attack by an opposing force (OPFOR) company. Subsequently, the infantry sends a request for fire support to a tank company.

5.1.2.4 Relevance to Federation Testing

Although MSCs are a way to describe the behavior of real-time systems, one MSC cannot describe the complete behavior of a system. It should be assumed that a federation will have a collection of MSCs in order to describe a testable behavior.

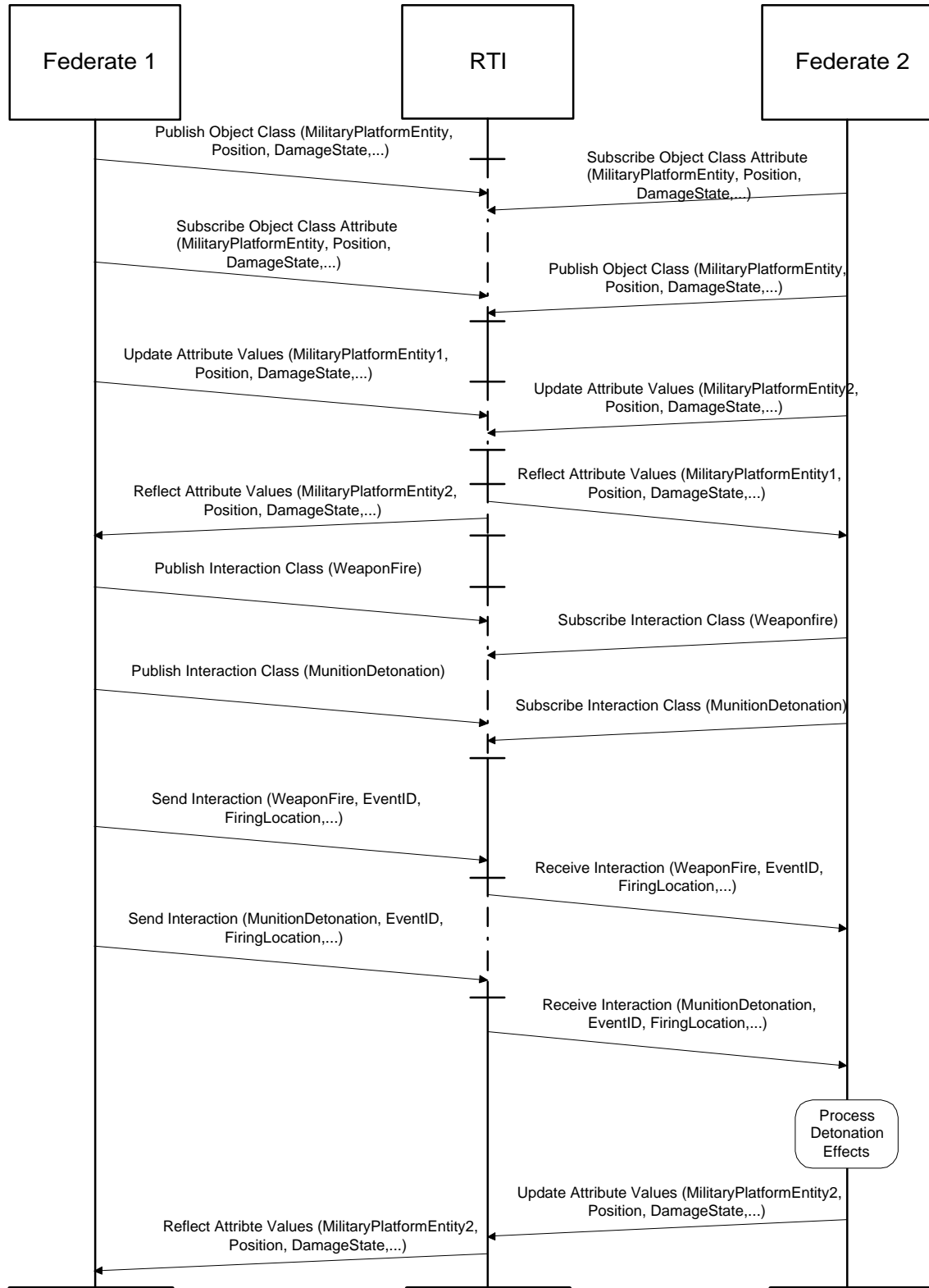


Figure 5.1. OIP in MSC Format for HLA Interface Services

When federations become involved in some higher level tests, such as Scenario Testing, the challenge will be determining what needs to be tested. An OIP described through an MSC will give a simulation tester an unambiguous way to test a simulation's behavior. The next step is to determine where in the federation specifications the OIPs belong: as an addendum to the FOMs/SOMs, as part of the FPW, or maybe as a stand-alone product for federates. More information about the use of MSCs can be found on the Web at <http://www.win.tue.nl/cs/fm/sjouke/msc.html> (Sjouke, 1997).

OIPs can be useful at several stages of Federation Testing, from Application Testing through to Scenario Testing. An adopted convention to describe testable behaviors is needed from the testing community. Only by using OIPs will automating the latter stages of testing be successful. The more that behaviors are defined, the easier the job of automated testing becomes, and therefore the more complete and correct the conducted tests can be.

5.1.3 Federation Agreements

Federation agreements contain the high-level objectives of the federation and answer questions such as why the federation exists and what its goals are. Federation agreements can also describe federation requirements that do not fit into the FOM or FPW. This section contains a description of information similar to that contained in the FPW, but at a higher level.

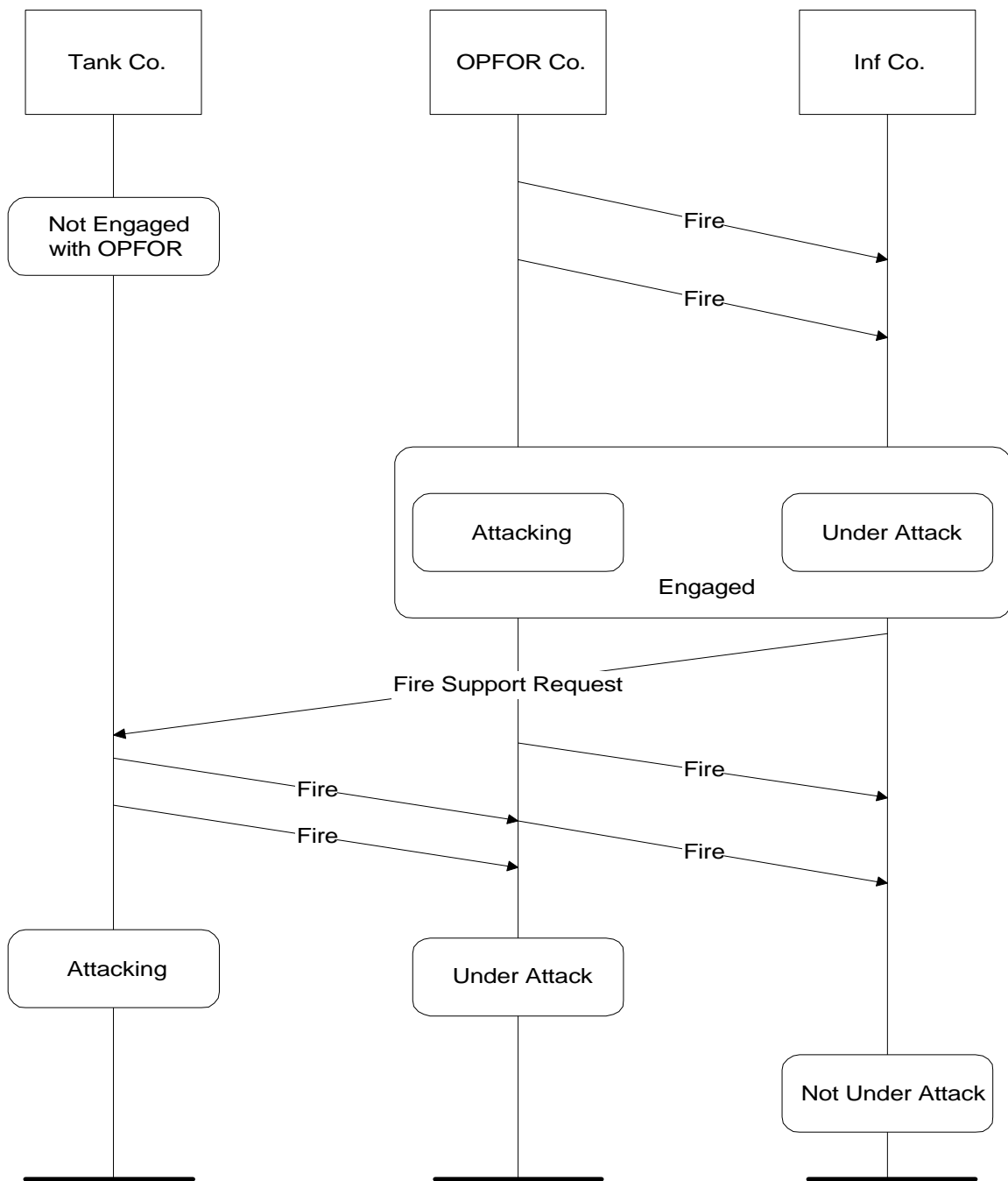


Figure 5.2. OIP Example for Scenario-Level Information

One of the important issues to be addressed in future efforts is: how can these agreements be captured in a computer-readable state? This is necessary if the data is to be used in automated testing. The FPW, described in a later section, sets a good precedent for this data capture. Future efforts should look at capturing high-level requirements in a testable format. See Section 5.2.3 for a discussion of a format for test procedures. Also see the Graffagnini SIW paper for additional ideas on automating the testing of federation requirements. (Graffagnini, 1997)

5.1.3.1 Overview

Scenario information defines what a particular federation execution instance will do, including which objects and how many will participate and what configuration parameters will be used.

At one level, federation capabilities define whether a federation can pause and resume, save and restore, and transfer and accept attribute ownership. On another level, it also includes assignments of some common simulation functionality (e.g., data logging, performance monitoring, simulation monitoring) to specific federates. For example, there may be two federates capable of logging a certain data set, but only one is assigned this responsibility for a particular execution. These kinds of capabilities are important to consider during federation development to make sure that development efforts are not redundant and that all desired functionality is included in the resulting federation. Note that choices for inclusion of capabilities and assignments to specific federates must be made with the big picture and the timing model for the federation in mind.

While the FOM defines a significant portion of a federation's data requirements, there are additional federation execution requirements that need to be captured. These additional Federation Agreements typically fall into the following areas:

- execution environment
- runtime performance
- data representation
- timing issues
- data semantics
- scenario description
- federation capabilities

Some of these categories are discussed in the paper "High Level Architecture (HLA) Performance Framework" (Dahmann et al., 1997), which was presented at

the Fall 1997 Simulation Interoperability Workshop (SIW). Execution environment agreements answer questions such as:

- What hardware is needed?
- What type of network configuration is being used?
- How many computers are being used?
- What platform architectures are being used?
- Where are the computers located?
- Who will be operating them?

Runtime performance agreements formalize the runtime requirements for federation execution. They answer questions such as:

- Is real-time execution required?
- If so, what latency can be tolerated?
- If not, what are the upper and lower bounds on execution time for the federation?
- Also if not, what are the upper and lower bounds on execution time for each federate?

Neither the Interface Specification (IFSpec) nor the Runtime Infrastructure (RTI) API dictates the representation of data transported via the RTI. This decision is left to each federation developer and should *always* be documented, both to decrease the possibility of error and to allow testing of each federate to the data representation specification. For the character string data type, the character set should be specified (e.g., ASCII), the choice of null termination versus length specification must be made, and any length restrictions should be indicated. For all other data types, it is best to formally document the representation of each data type on a byte-by-byte level. This step eliminates the possibility of incompatibility due to platforms, compilers, or assumptions made by developers (provided, of course, that each federate developer adheres to the documented representation). Even if the choice is made to not support heterogeneous federation execution, this choice and the rationale for it should be documented.

There are many time management parameters for a federate, including regulation, constraint, lookahead, and the relationship between internal and federation clocks. The choice of time management parameters for a single federate will often affect the overall performance of the federation. As such, time management needs to be addressed in federation design. Dependencies between the federates should be documented. Some time management issues are related to runtime performance requirements. For example, hardware-in-the-loop federations must consider how to monitor the performance of the software components and how to respond if performance is not satisfactory. Non-critical

software components can be terminated or ignored, but critical ones cannot. Other timing issues are related to data semantics. Some of these issues, such as update conditions, already are addressed in the FOM. While timing requirements per data item are indeed useful, there also is a need for a big-picture understanding of how time is understood, implemented, and managed for a particular federation.

The FOM includes quite a bit of information regarding the data items to be transported in a federation. However, the criticality of the description field for each data item is probably not as well understood as it should be. The meaning of each object, attribute, interaction, and parameter must be defined in enough detail for every participant in the federation, and the level of detail and content required for this description varies among federates and application domains. For example, an analytic or engineering-level federate may require knowledge of how certain data items are used in equations implemented in other analytic/engineering federates, but a training federate has no need for this type of information. Most federates also need to understand the big-picture concept of how the data items in the object model are intended to work together. This concept was described in Section 5.1.1.

5.1.3.2 *Current State*

The concept of Federation Agreements was originally discussed in the FEDEP as the Federation Required Execution Details (FRED). The Federation Execution (Fedex) Planning Workbook (FPW) is now being defined as the repository for some of this type of information. The initial FPW definition addresses the categories of execution environment and runtime performance in detail and includes some scenario information in the object/interaction tables. See Section 5.1.4 for more information about the FPW.

5.1.3.3 *Relevance to Federation Testing*

Federation Agreements are essential to Federation Testing. They serve as the requirements to which the federation should be tested and they can allocate certain requirements to specific federates in the federation. In some cases, the latter function can decrease testing costs by allowing some tests to be performed on a subset of the federation. Each federate can be tested individually to verify that it upholds the Federation Agreements. In addition, when a federation fails Federation Testing, documented agreements can help the correction process.

Documentation also makes it possible to perform more tests, which, in turn, increases federation reliability. In addition, when the documentation is in a shareable electronic format, it allows automation of the testing process, which, in turn, increases efficiency and makes it possible to perform more tests in the same, or shorter, period of time. An electronic format that demonstrates the

dependencies between agreements also will make it possible to update these agreements automatically during federation development.

Another aspect of the testing process that should be emphasized is proper management of the federation requirements. If federation participants understand the requirements up front, fewer errors will be found during testing and "real" exercises. Also, the organization (presentation) of these requirements will affect their understandability, hence their utility. Although more agreements should be documented (since they need to be tested), information overload can be as much a problem as lack of information. One way to address this issue is for the Federation Agreements format (or formats) to make critical and frequently used information readily available, with more information accessible as desired.

There is an effort in progress to define a DIF for the FPW, and the Federation Testing community certainly should be involved in this definition.

5.1.4 Fedex Planning Workbook (FPW)

As has been discussed in previous sections, the FPW is one of the data requirements for the automations of Federation Testing. This section describes the FPW in more detail.

The FEDEP version 1.1 replaces the earlier concept of the Federation Required Details (FRED) with a Fedex (Federation Execution) Planning Workbook (FPW) to document critical information needed to run a Fedex. The FPW format was defined by a DMSO-sponsored RTI Performance Technical Exchange in May - July 1997. The initial implementation of the FPW is in a Microsoft Excel™ workbook, but DMSO is planning to fund the development of a new tool to create and manage the FPW in the near future.

5.1.4.1 Description

The FPW is a set of five tables for specifying the performance characteristics a user needs from an RTI implementation. It also facilitates communication about a federation execution by reducing ambiguity in execution details that affect performance.

The five tables (or sheets in the Excel workbook) are as follows:

1. Federation Execution Summary Table: defines at a high level the composition of a federation execution; used to describe execution details of federation, including:
 - federation execution name

- member federate information
 - name
 - API used
 - tick rate
 - time regulating and constraining status
 - host and LAN that federate is executing on
 - version of RTI software
 - number and name of concurrent federation executions
2. Host Table: provides details about hardware that affect performance of federate and RTI; used to describe the following information for each host:
- hardware
 - architecture
 - number of CPUs
 - operating system
 - free memory available to RTI
 - % of CPU available to RTI and federation
 - % of CPU available to RTI
3. LAN Table: provides information about bandwidth availability and latencies introduced by network infrastructure; used to describe the following:
- each LAN in the federation execution
 - physical type
 - throughput available to Fedex
 - LAN to LAN connectivity
 - type of device used to connect each LAN
 - effective throughput available to Fedex for each device
 - latency introduced by device
4. RTI Services Table: lists the services a federation execution uses. The services used may affect the performance characteristics of a federation execution. This table is filled out once for each federation execution, specifying for each RTI service in the current IFSpec whether that service is used at least once in the federation execution.

5. Object/Interaction Table: specifies runtime characteristics related to FOM data that affect performance of federation execution; one table is filled out for each federate, listing:

- objects
 - number simulated by federate
 - attribute sizes, nominal and maximum update rate, maximum tolerable latency, attribute transport and ordering, update groupings, ownership transfer groupings
- interactions
 - interaction transport and ordering
 - parameter sizes, nominal and maximum update rate, maximum tolerable latency

5.1.4.2 Relevance to Federation Testing

Ultimately, when an FPW DIF is defined, the workbook can be used to drive Integration and Application Testing. In the meantime, federation testers should be prepared to work closely with the contractor developing the FPW tool, to provide input on desired DIF characteristics and to maintain access to the latest workbook developments.

Like Federation Agreements, the FPW will become a necessary component to Federation Testing. During a test, many of the "other" aspects to test for, such as latencies, RTI performance, federate performance, will be tested against information contained within the FPW. Because of the relationship between Federation Agreements and the FPW, a certain test procedure will derive from either or both of these documents. An effort is needed to use the information contained in the FPW and design a set of test procedures associated with it.

5.2 METHOD REQUIREMENTS FOR FEDERATION TESTING

This section describes several key methods that should be used to facilitate the automation of Federation Testing. The MOM is discussed as a way to manage testing. Then, the importance of standardizing file formats is described. One file format that the FTS team has introduced is one for test procedures. Additionally, standardized file formats are discussed as a means for integrating separate testing and HLA development tools. Finally, the use of the internet to facilitate testing is discussed.

5.2.1 Use of the Management Object Model (MOM)

The Management Object Model is a special kind of FOM for the management functions of the RTI. Previously the MOM was described in its own separate document for the HLA Version 1.1 specifications (DMSO, 1996d), but starting with the Version 1.3 specifications, the MOM is part of the Interface Specification and eventually will be included in the IEEE standard for the HLA. The MOM is an auxiliary component of the RTI and must be present in every federation. The MOM classes, with their pre-defined attributes and parameters, are used by the RTI in some of its internal functions as well as allowing a separate manager federate to monitor the status of the federates or to control a federate's behavior (Hyon & Seidel, 1997). The MOM uses the same mechanisms for the management of a federation that the RTI uses for the exchange of information among federates.

The MOM differs from the normal HLA data exchange mechanisms in several ways:

- MOM classes, attributes and parameters are pre-defined for every federation.
- The RTI is responsible for publishing, subscribing to, and generating many of the objects and interactions. Normally, no federate involvement is required.
- A manager federate need only subscribe to the Manager object class and subscribe to and publish appropriate Manager interaction class to have full access to understanding and control of the member federates of a federation.

Basic MOM actions include the following:

- Generate interactions when something interesting happens
- Respond to control interactions that modify the state of the RTI
- Create management objects on behalf of the federate and federation
- Update attributes of management objects in a desired way
- Respond to Request Attribute Update for management object attributes
- Allow federation designers to extend the basic MOM

5.2.1.1 MOM and Federate Compliance Testing

MOM objects and interactions are used to log all the interaction services between a federate and the RTI in testing the federate for HLA Compliance. This service interaction log is used to verify that the federate actually performs the actions

required by the interface specification and the federate's conformance statement. See (McLean & Loper, 1997).

Figure 5.3 shows the implementation of a Logger federate, which subscribes to MOM interactions to record service interactions between a Federate Under Test (FUT) and the RTI. Once these interactions have been logged, the log file is examined to confirm HLA compliance for the FUT.

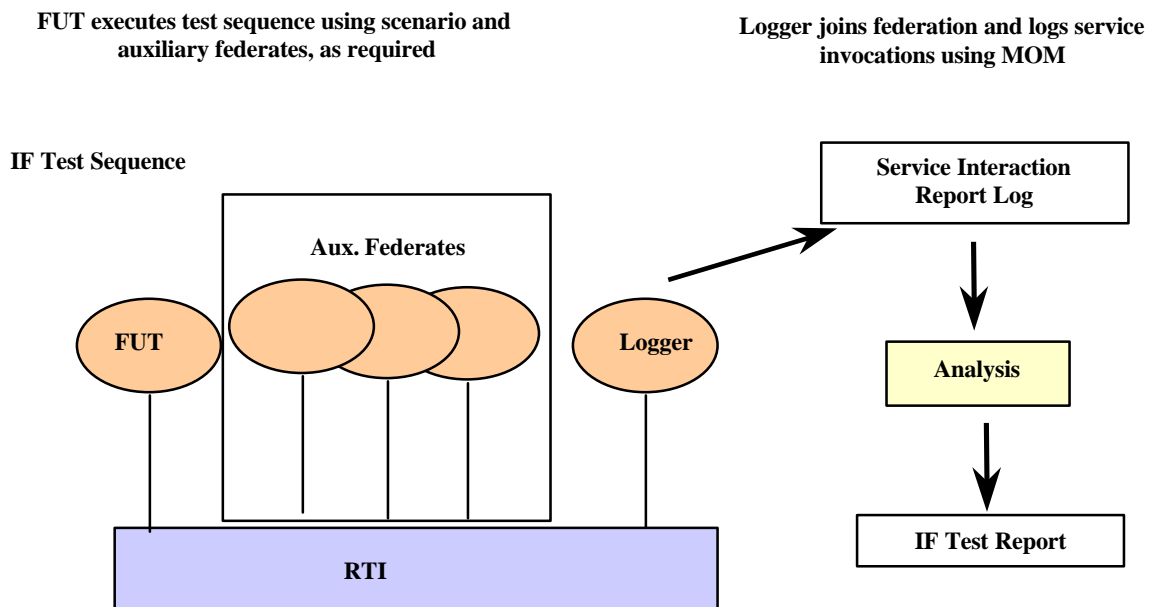


Figure 5.3. Use of the MOM for Federate Testing

5.2.1.2 MOM and Federation Testing

The utility of the MOM for federation testing increases with Version 1.3 of the HLA as the MOM becomes part of the Interface Specification and formally a part of the proposed IEEE standard for HLA. Federation test tools can now rely on a standard implementation of the MOM with every RTI implementation, so they can take advantage of MOM features to log RTI service interactions to verify test objectives. Research is needed to reduce the impact of logging on federation/RTI performance (bandwidth, latency), to increase understanding of distributed logging and log synchronization, and to develop smart filters to determine what to save in a log file for later processing in an after-action review vs. what could be saved in a reduced file after processing during run time.

5.2.2 File Formats

The standardization of file formats will be an important enabler for the success of automated federation testing. The more that data describing the federation can be standardized into interoperable file formats, the more sharing of data and therefore automation of testing can be facilitated. The following sections describe some of the existing types of file formats that are found in distributed simulation (specifically: ModSAF and JMASS file formats.) The goal of this section is to provide a background for future efforts, so that the issues associated with many different file formats will be dealt with proactively, especially with regard to federation testing. One of the answers to the proliferation of multiple file formats is the Data Interchange Format or (DIF). DIFs are described in Section 5.2.2.3.

5.2.2.1 ModSAF RDR

ModSAF has many data files that utilize the "libreader" format at runtime to initialize this simulation. Libreader provides a facility for reading data files into C structures. A typical .rdr data file consists of "an arbitrary number of data items." [LM, 1997 #43]. Data items that libreader can support are specified in the ModSAF libreader.info file. The idea behind libreader is to have a structured syntax so a file can be read into one data structure. (See Section 9.1 for the format of the data structure.)

Thus, libreader can read in an integer (stored as a four-byte signed integer), a real number (stored as a four-byte float), symbols (stored in a symbol table), and an array. The array is stored as a sequence of contiguous four-byte `READER_UNION`'s. The first element is always of type integer and indicates the length of the array (including the 0th element). The remainder of the elements follows at indexes [1] through [length-1]. Arrays are denoted by the use of parenthesis, and lines in the .rdr file that begin with ";" are ignored. (See Section 9.1 for an example of a .rdr file (routemap.rdr) taken from ModSAF.)

5.2.2.2 JMASS File Formats

The Joint Modeling and Simulation System (JMASS) is a modeling and simulation (M&S) system developed by the Air Force for use by the Acquisition M&S community. This system provides software infrastructure, a family of data file formats, and a set of tools to standardize and simplify the development and interoperability of models, as well as the execution and post processing of simulations using these models. JMASS defines data file formats for several dozen types of files used within and/or with the JMASS system. These data file types and formats provide standard ways to exchange information between different tools in the system. They also provide a common way to get data into and out of JMASS models during a simulation execution.

Of particular interest are the family of file types and formats used to configure a simulation execution (and its component models) and journalize its results. It includes simulation, scenario, team player instance, configuration, override, journal metadata, ASCII and binary simulation journal data, antenna pattern, and route file types. This family of file types is dependent on the software infrastructure provided in JMASS, which capitalizes on the knowledge that data-driven models are more flexible and that analysis models and simulations are typically data-driven. A standard mechanism for developing data-driven models can reduce model development time (since a significant amount of code is provided in the form of interfaces and libraries), reduce model maintenance efforts (since data-driven models can be modified for many purposes by providing new data sets), and reduce the learning curve for operating individual models (since the same types of data files are used in all JMASS models regardless of which government agency and contractor developed the model).

Execution of a JMASS simulation requires a command line argument specifying the name of the simulation file. The simulation file specifies the scenario, team player instance, ASCII metadata, and binary metadata files. The scenario and team player instance files jointly define which models will participate in the scenario and on which host computers they are to be executed. These files are read (by the JMASS-provided main thread of execution), specified simulations are started, and specified models are instantiated in the simulations. Also in the scenario file, a configuration file is specified for each model instance. In turn, the configuration files are read and initial values for data items for each model are set, as specified in the configuration file, using the software interfaces and infrastructure provided. The override files are used in a second round of initialization (overriding initial data values from the configuration file with values specified in the override file). This step allows use of a standard set of configuration files that define the most commonly used values for most cases. Only values that make a particular instantiation of the model unique must be rewritten in the override file.

Another noteworthy aspect of the JMASS file format suite is the common definition format and the common trends among the formats. (Similar issues probably are being addressed in some of the data standardization research being conducted for DMSO.) The formats are all defined using an extended BNF. Almost all of the data files are stored in ASCII format. Most allow comments beginning with two consecutive dots ".." and continuing to the end of the line (similar to "--" in Ada and "//" in C++). Many use the concept of "keywords," which are specified by a leading dot "." character. Keywords divide the data files into sections that logically follow the makeup of the simulation and/or models that are being parameterized. An example of a configuration file, which provides input data for a model and its components, is shown in Table 5.1.

```

..UNCLASSIFIED
..Example configuration file

.class AFF200Player, "Configuration data for the AFF 200"

.component_instance SignalProcessor
    .component_instance PowerDivider
        LossDb                = 10.0
        NoiseDensity           = 1000.0 hz
        NoiseControlSwitch     = ON
    .end

    .component_instance DopplerFilter
        CutoffFrequency        = 20000.0
    .end

.end

```

Table 5.1. Example JMASS Configuration File

5.2.2.3 Data Interchange Formats (DIFs)

Part of the simulation community's commitment to move to the HLA is through its use of data standardization, particularly Data Interchange Formats (DIFs). Examples include the establishment of the Object Model Template (OMT) DIF, which is motivated by the need "to provide an unambiguous way to exchange OMTs used to describe FOMs and SOMs," which are key to the issue of reuse (Scruder & Sheehan, 1997).

The OMT DIF is defined in Bacus Naur Format (BNF) using ASCII text. The following section is taken from documentation on the OMT DIF and is repeated in Table 5.2 below for easy reference (DMSO, 1997d).

Certain symbols within the BNF have special meanings, these are called meta-symbols and they are used to structure the BNF. Double Quotes, Angle Brackets, Braces, etc. are meta-symbols within BNF, and their definition and use will be given below.

Words inside double quotes ("word") represent literal words themselves (these are called terminals). In addition, terminals will also be highlighted using boldfaced text. An example of a terminal is "HLA-OMT".

Words contained within angle brackets '<>' represent semantic categories (i.e. non-terminals) which must be resolved by reading their definition elsewhere in the BNF. An example of a non-

terminal is <NameCharacter>.

The BNF used in this document adds a special case of non-terminal which is denoted by double brackets '<<>>' rather than single angle brackets. This special case non-terminal is a reference to the DIF Meta-Model, and further details about the non-terminal, including its definition can be found in the data model glossary.

A production rule is a statement of the definition of a non-terminal. It is designated by the production meta-symbol '::=' which assigns the definition to the right hand side (RHS) of the production to the non-terminal on the left hand side(LHS) of the production symbol. The LHS must always consist of a single non-terminal, while the RHS can consist of any combination of terminals and non-terminals. An example of a production rule is:

```
<Integer> ::= 0..65536;
```

Which defines the non-terminal <Integer> to be a number between 0 and 65536.

Optional Items are enclosed by square bracket meta-symbols '[' and ']'. Square brackets indicate the item exists either zero or one time, that is to say, it may or may not exist. An example of an optional item is [<Sponsor>] which indicates the Sponsor item may or may not be present in the DIF.

Repetition (zero or more) is performed by the Curly Brace meta-symbols '{' and '}'. Curly braces indicate that there may be zero, one, or more sequential instances of the item. An example of curly braces is {<Version>} which indicates there may be zero, one or more versions present in the DIF. In basic BNF iteration is described using left or right recursion, extended BNF uses the curly brace meta-symbol to make reading the BNF easier. An example of how the curly brace is used to replace recursion is given below:

The Rule:

```
<ident> ::= <Letter> { <Letter> | <Digit> }
```

is the same as the recursive rule:

```
<ident> ::= <Letter> | <ident> [<Letter> | <Digit>]
```

The double period .. used within a Literal is a shortcut notation for denoting the set of ASCII characters between the characters to either side of them. An example of this is "a..z" which denotes the set of lowercase letters between 'a' and 'z' inclusive.

All BNF Statements are terminated by a semicolon ;

In order to represent unprintable or reserved characters within the BNF and the DIF code itself, we use the C language convention of using the backslash as the escape character within literals with the following set of legal escaped literals:

'\n'	newline
'\r'	return
'\"'	Single Quote
'\"'	Double Quote
'\\'	Backslash
'\t'	Tab
'\b'	Backspace
'\f'	FormFeed
'\xxx'	Any character where xxx is its Octal representation.

Table 5.2. OMT DIF Description

Tools have been developed to test for data consistency using the DIF specifications, which represent one of the foremost tests during Federate Testing. It also should be noted that the Federation Execution Details (.fed) file format has also been standardized using this DIF notation.

5.2.2.4 Self-Describing Data Interchange Formats (DIFs)

One problem with DIFs, however, is that the standards are still evolving. Programmers must stay abreast of all modifications, then change their applications accordingly. Currently, there is an effort underway at GTRI to tackle this issue by developing self-describing databases and database generators. By combining these new systems with formal grammar definitions of DIFs and self-describing data messages that contain identifications of the DIFs used in the messages, researchers believe they can help preserve interface investments during the gradual and moderate evolution of DIF standards.

A self-describing database is a hierarchical database in which data at some level is either an extension of the data above or an intension to the data below. The highest level of the database consists of meta-data that describes the data model in terms of itself, hence the term "self-describing." The top layer, referred to as meta-schema, contains a dynamic data dictionary used to support access to the meta-data needed for interchanging automatically interpretable data between

databases. In other words, this data dictionary describes the semantic content of the grammar to be used to store the data.

A database generator system supports a self-describing database of sets of objects defined by grammars. It accepts input in the form of context-free grammar that defines a set of objects for which database support is desired. From the grammar, it generates a database schema definition, a parser to parse and populate the database with objects defined by the grammar, and a set of equations for query computation on the objects stored in the database. A database generator system supports multiple levels of data decomposition to facilitate efficient analysis of objects stored in the database. A database generator also can automatically generate syntax-directed editors from grammar specifications, thereby supporting descriptions of objects that are acceptable in the new grammar. In addition, it can support syntax-directed query formulation on and across objects stored in the generated database. A database generator can assist in composing/decomposing objects according to sub-grammars, which could help define mappings between objects with different decomposition levels in a database. Ultimately, a database generator could even use its self-describing meta-schema to provide syntax-directed assistance in the initial definition of a grammar.

5.2.3 Test Procedures

One challenge to simulation developers will be defining test procedures for a federation. As HLA-compliant federations are put together, questions of what exactly needs to be tested will arise. The solution to this problem will be the development of a test procedures library that other federates can access to develop their own test procedures. In addition, for this process to be successful, a standard format needs to be established for test procedures. One possibility is to apply a DIF.

5.2.3.1 Test Procedures Format

During the development of the Federation Testing System (FTS), the Integrated Development Team (IDT) developed an internal test procedure format. The team focused on adopting a format that would meet the needs of the user community and would be easy to input into the FTS. This format uses a hypertext format and is shown in Table 5.3.

```

<TP> [TP Group Reference] [TP Group Title]: [TP ID] [TP Reference] [TP
Title]

<INIT_COND>
[Initial Conditions Text]
</INIT_COND>

<REQS>
[Requirements Text]
</REQS>

<CAPS>
[Capabilities Text]
</CAPS>

<SCENARIO>
[Scenario Text]
</SCENARIO>

<CRITERIA>
PASS: [Pass Criteria Text]
FAIL: [Fail Criteria Text]
</CRITERIA>

<PAGE>

<TITLE>
[TP Title] - Page [Instruction Number]
</TITLE>

<FUT_INSTR>
[FUT Instruction Text]
Press INSTR COMPLETE.

- OR -

Press PASS if [Passing Criteria].
Press FAIL if [Failing Criteria].
</FUT_INSTR>

<TSTFED_CUE>
[Test Federate Script Filename]
</TSTFED_CUE>

<ANALYSIS_CUE>
[Analysis Federate Script Filename]
</ANALYSIS_CUE>

```

```

<SUBJECTIVE>
[Subjective Subtest Title]
</SUBJECTIVE>
</PAGE>
</TP>
<END>

```

Table 5.3. Test Procedure Format Sample

5.2.3.2 Test Procedures Description

As noted above, one of the challenges in testing a federation is determining what needs to be tested. Initially, developing test procedures may prove to be a slow and arduous task. However, if a library of test procedures from existing or previous federations is developed, new federations can borrow them to help develop their own test procedures. For this idea to become a reality, a standardized format for exchanging test procedures needs to be accepted. For the FTS, a Test Procedure Set was developed to document and organize the test procedures. The format described in this document provides the necessary elements to read in and execute a Test Procedure Set with the Test Session Manager. The structure of this format supports the orchestration and synchronization of testing events between the FUT, the test federate, and the analysis federate (described in Section 6.3) during test procedures. Test procedures are conducted through the execution of FUT instructions, and testing events are coordinated by their relationship to the instructions.

5.2.3.3 Test Procedures Example

A detailed example of the test procedure format can be found in Section 9.3. This procedure currently is being used in the FTS as an example of a test procedure for the RPR FOM. Standardizing the test procedures format will allow other federations with similar FOMs to reuse existing test procedures.

5.2.4 Integration of HLA Tools Developed By Different Organizations

Clearly there is a need for standard data interchange formats to support federation testing so that automated tools can be developed by different organizations which are interoperable but tailored specifically to address the needs of different federations or different stages in the development of federations.

Figure 5.4 illustrates the relationships between the Object Model Development Tools, data interchange formats, and federate/federation testing. The OMDT can import data elements from the Object Model Data Dictionary (OMDD) in OMDD DIF files, and then generate OMT DIF files for federate testing and FED DIF files to support federation testing. The RTI Service Table from the FPW can be used in both federate and federation testing, while the Object/Interactions Table can provide scenario data for federation testing. MOM data available from the RTI supports both federate and federation testing.

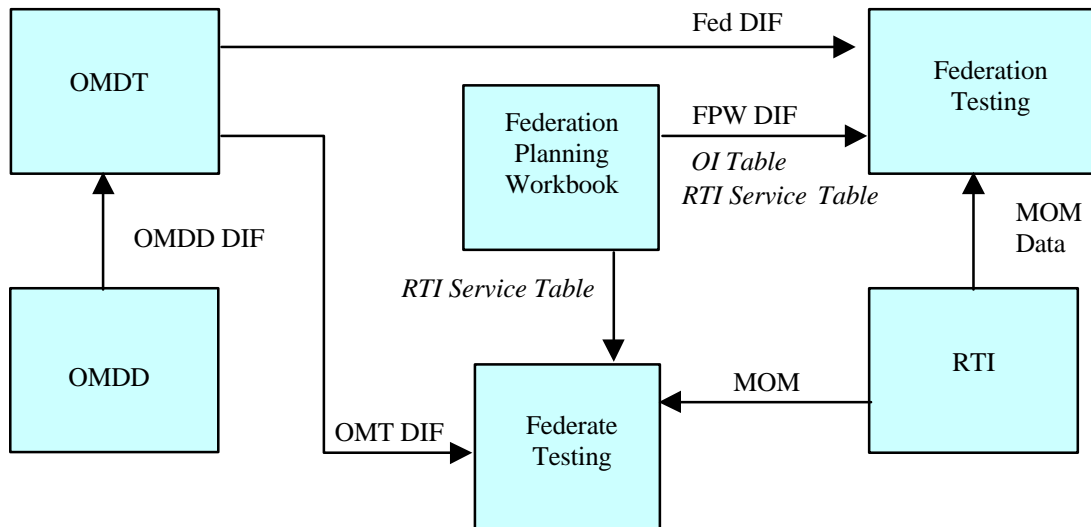


Figure 5.4. HLA Object Model Integrated Tool Suite and DIFs Support the Test Process.

5.2.4.1 Object Model Development Tool (OMDT)

The OMDT provides a mechanism for developing SOMs and FOMs, either from scratch or by compiling data elements from the OMDD. The OMDT was developed in direct response to comments from the first HLA protofederations that object model development using spreadsheets was cumbersome and tedious. By storing all object model data in one database, the OMDT guarantees consistency among the various OMT tables. In addition, the OMDT includes a Consistency Checker that can be used to verify that the SOM or FOM conforms to the OMT Specification. The Consistency Checker provides the first level of federation testing by ensuring that the FOM conforms to the OMT Specification.

5.2.4.2 FPW Development Tool

When an FPW DIF is developed, it will provide a parseable mechanism for automating standardized tests of federation agreements and object interaction protocols. The RTI Service Table will specify which interface services the federation can respond to and invoke, and the Object/Interactions Table can indicate which federates supply and consume what data. Federation test developers need to coordinate with the FPW tool developers to provide testing requirements and suggestions, especially in the area of tailoring the OI Table to yield OIPs in some automated fashion. The FPW DIF is important because it provides the parseability necessary for automation.

5.2.5 Use of the Internet to Facilitate Test Management

With the immense popularity of the Internet, organizations have started to accept the World Wide Web as a required tool to support widely dispersed working environments. As such, the Web should be used actively and with vision to support the requirements of the Federation Testing Process.

The HLA Federate Conformance Test System, <<http://hlatest.msosa.dmsso.mil>>, provides a good example of the ways in which the internet can offer many advantages to both test users and test system administrators. Perhaps the key advantage is the availability of a common and easily accessible means to communicate. For example, many test management functions such as submitting test requests and reporting test results lend themselves to a World Wide Web-based system. Browsers, common gateway interface (cgi) scripts, and underlying data structures can be assembled to create a complete system that controls test operations from start to finish. Because of the common availability of these components and their universal usage, such systems allow users and system administrators to communicate and conduct test operations without the need for special software and hardware.

5.2.5.1 Federate Conformance Testing Example

The HLA Federate Conformance Test System was developed for DMSO for providing an easily accessible system to conduct and administer federate HLA Conformance testing. As shown in Figure 5.5, the Federate Conformance Test System combines HTML, cgi scripts, automatically generated e-mail, databases, and federate test tools into one comprehensive system. This system is illustrated in three layers where, from top to bottom, the test user (i.e., customer), test system infrastructure, and test system administrator (i.e., operator), respectively, work within the same framework to conduct all test operations.

To demonstrate the comprehensive nature of such a system, the following sequence of events represents key exchanges central to test operations:

Initiation: The customer accesses the Web-based test system to request a test for a given federate. An application form is submitted, an e-mail message is generated automatically to notify the operator of the application, and the application data is written to the request database.

Approval/Acceptance: After receiving the e-mail notification, the operator reviews the customer's application using a browser and ensures that all data are complete and accurate. If the information submitted is correct, the operator approves the application and automatically generates an e-mail that provides further instructions for conducting the test (e.g., prompts for providing a Conformance Statement, OMT file, and other test files.)

Customer Provides Test Files: Using a browser capable of multi-part form uploads, the customer uploads the requested files. The file upload automatically generates an e-mail notifying the operator that the files have been received, along with an email to the customer confirming the file upload success.

SOM Conformance Tests and Results: Once all of the necessary files for testing have been received, the operator conducts the SOM conformance test and the Conformance Cross Check test. When these tests are successfully passed, the operator generates a test sequence for the Interface Test. The operator sends an email to notify the customer of the successful test results and to send the customer the test sequence for the Interface Test and instructions for providing other information needed for the IF Test.

Customer Provides Test Files: Again the customer uploads the requested files and/or provides interface test information through web based forms. A date and time are confirmed for the IF test by both the customer and the test operator using email.

Interface Test Operations and Results: The IF Test is conducted over the Internet. The customer starts the RTI, creates a test federation, and the FUT joins the federation. The test operator then joins the federation with a special logger federate. The FUT then exercises the test sequence while the logger federate uses the MOM to log all RTI service calls and responses from the federation. After the test is over, the test operator examines the test log to verify that the FUT successfully demonstrated the test sequence. The customer is notified of the test results by email.

Through this back and forth exchange between customer and test operator, all tests are conducted, administered, and HLA compliance is established through the use of this single Web-based system.

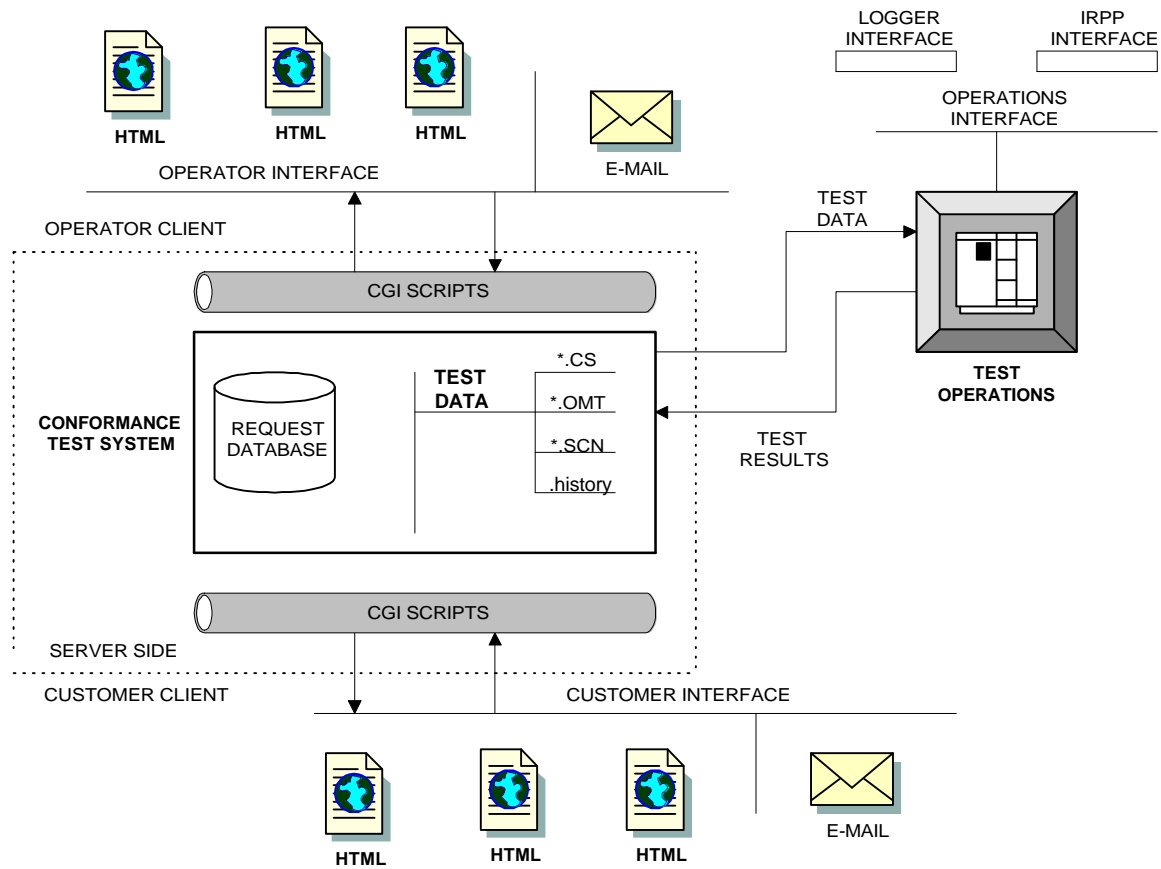


Figure 5.5. Overview of the Web-Based Federate Conformance Test System

5.2.5.2 Implications for Federation Testing and Future Directions

The use of the internet will be most useful for Federation Testing, when the federation developers, integrators, or users are located at several distributed locations. In these cases, internet communication, facilitation, and also distributed Federation Testing will prove to be necessary. This is further evidenced by recent uses of the Internet to store, manage, and maintain central repositories of information for access by the HLA community. An Internet and Web based federation test system will provide more flexibility than a locally based system.

One example of a good approach to distributed test management is highlighted in the Morley and McLean paper from the 13th DIS Workshop, entitled "Issue Management Database System." (Morley & McLean, 1995) This paper describes the successful use of web-based technologies for distributed Integration, Functional, and Scenario testing and should be a required capability for all future large-scale testing environments.

This Page Intentionally Left Blank

6. TESTING TOOLS

Many testing tools have been built to support distributed simulation testing since the time when two simulators were first linked together up until the current HLA testing tools. The overarching reason for tool development is because distributed simulation testing is hard.

It is well known that desktop simulation verification is difficult. (Law & Kelton, 1991) The troubles with software quality resulting from insufficient software testing are also well documented. (Schach, 1990) Add to the mix the complexity resulting from developing a distributed system (Schach, 1990) and the concept of distributed simulation testing could be given up as being *impossible*. Unfortunately, this is the approach taken by many distributed simulation developers: "It's too hard, this distributed simulation system can't be tested so lets just build it, do a face validation, and fix the errors we find along the way."

In spite of this view, successful implementers of distributed simulation for both Test and Evaluation (McKee, 1998) and Analysis (Roberts, 1995) have shown that systematic and practical testing and VV&A can provide a robust functional system that can faithfully represent the requirements of the sponsor.

One of the keys to making distributed simulation testing manageable is through the development of tools. If a test, test process, or test procedure can be characterized well enough so that it can be automatically done, then doing this will allow the testers to spend more time on the problems that have to be done by humans. (Clay et al., 1995) The ultimate goal is to increase the coverage of the testing so that a greater percentage of the required federation behaviors are verified. Automation is also key for regression testing. When software changes are implemented, if automated tests exist, then these can be run and comparisons made. This allows the federation developers and testers to do more effective change control.

The following sections discuss some of the important simulation testing tools that have been developed. The first section gives a brief history of some of the testing tools that were developed to support DIS and ALSP. The following sections discuss a brief history of distributed simulation testing, then details are provided on the HLA Conformance Testing tools and the Federation Testing System (FTS).

6.1 TESTING TOOLS FOR DISTRIBUTED SIMULATION

Many of the testing tools that have been developed for DIS can trace their heritage to the Institute for Simulation and Training (IST) Intelligent Simulated Forces CGF Testbed, which became the first defacto DIS Test Tool. This test

tool, which was converted from SIMNET to DIS, was used to test 41 systems for the first DIS interoperability demonstration at the 14th Interservice/Industry Training Systems and Education Conference (I/ITSEC). (Loper, 1993; Loper, Goldiez, & Smith, 1993a; Loper et al., 1993b; Shen, Loper, & Ng, 1992; Vanzant-Hodge, 1994) The CGF Testbed eventually became part of the IST DIS Testbed which also contained Data Logger/Playback and Scanner Analyzing Tools.(Vanzant-Hodge, Cheung, & Smith, 1994)

The next major testing tool effort was developed to support the Advanced Research Projects Agency (ARPA) War Breaker Program. This program was the first to use DIS 1.0 in a large distributed exercise (Zealous Pursuit) and the first to use DIS 2.0.3 in a large distributed exercise (Zen Regard.) Initially, the program made use of legacy SIMNET testing tools which had been converted to support the 2.0.3 draft version of the DIS standard. Tools to support the following capabilities were used:

- PDU/Protocol Verification
- Tactical Display
- 3D Stealth
- Network Traffic/Bandwidth Monitor
- PDU Generation
- Data Logger

As a result of the lessons learned in using these tools to support exercise integration, the DISIntegrator Integration Test tool was designed and developed. (Hansen, 1994) DISIntegrator was designed to be a real-time test tool that could analyze DIS PDUs while integration testing was being conducted, in order to facilitate faster detection, analysis, and fixing of DIS implementation errors. (Clay et al., 1995) DISIntegrator was the first test suite to do real-time dynamic testing of distributed simulation.

The DIS Test Suite (DTS) was the next major step in test tool evolution. The design of DTS built on the lessons learned from the IST Testbed, DISIntegrator, and testing to support the STOW-E exercise. (Long, McAuliffe, & Liu, 1996; McAuliffe, Long, Liu, Hays, & Nocera, 1995; McAuliffe, Long, Liu, Hays, & Nocera, 1996) The Federation Test System (FTS), discussed in Section 6.3 is based on the software framework designed for DTS.

The other major distributed simulation paradigm, the Aggregate Level Simulation Protocol (ALSP) also has a heritage of test tool development. See the Page and Babineau paper presented at the Fall 97 SIW, "The ALSP Joint Training Confederation: A Case Study of Federation Testing." (Page & Babineau, 1997)

Also see the ALSP web site: <<http://alsp.ie.org/alsp/alsp.html>> for more information.

6.2 HLA COMPLIANCE TEST TOOLS

Compliance Testing tools were developed to assist in the four-step HLA Compliance Testing process explained in Section 4.1.2. There are three major components to the test tools: the Pre-Processor, Logger, and Post-Processor.¹⁰

The federate under test (FUT) must submit a Simulation Object Model (SOM), a Conformance Statement (CS), and a .fed file, and it may submit a Scenario file. Once these files are received, they are used by the tools described below as follows:

6.2.1 Pre-Processor

6.2.1.1 Conformance Cross-Checker

The Conformance Cross-Check process is illustrated in Figure 6.1 below. For the Cross-Check test, the services asserted in the FUT's CS are matched to services implied by the FUT's SOM, to determine if the two specifications are consistent.

¹⁰ This section was adapted directly from the series of Compliance Testing papers published by GTRI at SIW and DIS workshops and the "Federate Test Tools Operator Guide, Version 1.1," prepared for DMSO (DMSO, 1997a).

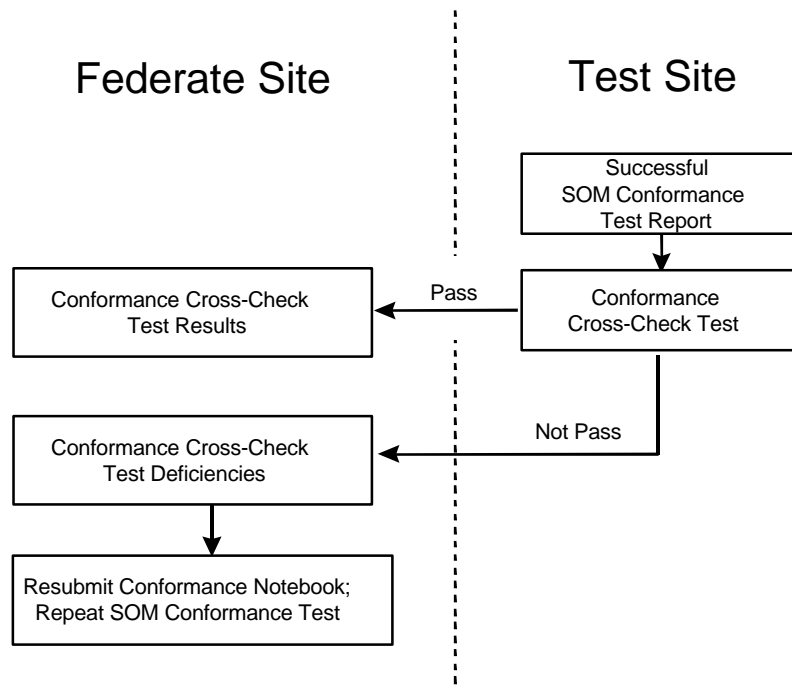


Figure 6.1. Conformance Cross-Check

6.2.1.2 Nominal Sequence Generator

The Nominal Test data ensures that the FUT can invoke and respond to all services for which it is capable, per its CS. The CS is compared to the Master Sequence to create the Nominal Sequence of services that the FUT can support. The Master Sequence is a dependency tree of all services defined in the Interface Specifications (IFSpec) (DMSO, 1997c). Where true dependencies exist (e.g., Publish before Update), the Master Sequence shows a mandatory ordering. If no dependency exists (e.g., Update and Pause), the Master Sequence ordering is arbitrary.

6.2.1.3 RepSOM Generator

A Representative SOM (RepSOM) is generated by using the SOM and the Scenario file (if provided). The RepSOM Test data ensures that the FUT is capable of invoking and responding to services using the range of data contained in its SOM. For example, a FUT may be capable of representing multiple objects, attributes, and interactions. Instead of attempting an exhaustive test using all combinations of the objects, attributes, and interactions, a subset of the SOM is chosen by the Certification Agent (CA) to represent the range of SOM data. This "logical subset" forms the basis for RepSOM testing.

A logical subset is derived from the SOM by a set of rules. These rules select one to three instances from each object, attribute, and interaction table. The

RepSOM Test data requires that the federate invoke the IF services specified in the Conformance Cross-Check Test associated with each Object Model Template (OMT) table for each instance requested.

6.2.1.4 Test Sequence Generator

The Test Sequence is generated by taking the Nominal Sequence and expanding it by the RepSOM. The FUT should be prepared to execute the Test Sequence multiple times within the test federation, as specified by the CA. The CA will log service interactions via the Management Object Model (MOM) interaction reports for later analysis and report generation (DMSO, 1996d).

6.2.2 Logger

Once the Pre-Processor files have been generated and checked, the CA will use the logger federate, which subscribes to MOM interactions to record service interactions between the FUT and the Runtime Infrastructure (RTI). The FUT and logger are connected via the RTI.

6.2.3 Post-Processor

The final step of IF Testing involves the post processing of the test logs. The Post-Processing tool is designed to reduce and analyze the service interaction log to determine whether each asserted service was demonstrated and whether the classes, interactions, and attributes specified in the RepSOM were demonstrated.

6.3 FEDERATION TEST SYSTEM (FTS)¹¹

The FTS Integrated Development Team (FTS IDT) is developing the Federation Testing System (FTS) under the sponsorship of STRICOM. The purpose of the FTS is to provide a software testing tool to perform HLA Federation Testing. Throughout the FTS development process, several FTS IDT meetings were held to define and document the user and technical requirements for the FTS, as well as for Federation Testing in general. (See Section 3 for a discussion of the steps taken by the FTS IDT to develop requirements for the FTS.) These early efforts produced two documents: the FTS Application Testing Process Requirements Definition (Acusoft, 1997a) and the FTS Application Testing Process System/Subsystem Design Description. (Acusoft, 1997b) These documents were

¹¹ Most of this section is adapted directly from FTS IDT, Acusoft, GTRI, TASC, and STRICOM documents describing the FTS, most notably the papers published at the 1997 Fall and Spring Simulation Interoperability Workshops.

the basis for the development of the FTS to support Application and Integration Testing, which is illustrated in Figure 6.2.

This Application Testing Process is broken into three phases, the Pre-Test phase, the Test Conduct phase and the Post Test Analysis phase. The concept of this testing process is to facilitate the creation, execution and analysis of Test Procedures for Application Testing. During the Pre-Test Phase the FTS provides tools to create Test Procedures and their supporting data elements and scripts. The output of the Pre-Test tools will be used later in the Test Conduct phase to execute the Test Procedures. FTS tools for the Test Conduct phase consist of the Test Federate, Analysis Tool and Test Session Manager. During the Test Conduct phase, the FUT will communicate with the FTS via the RTI.

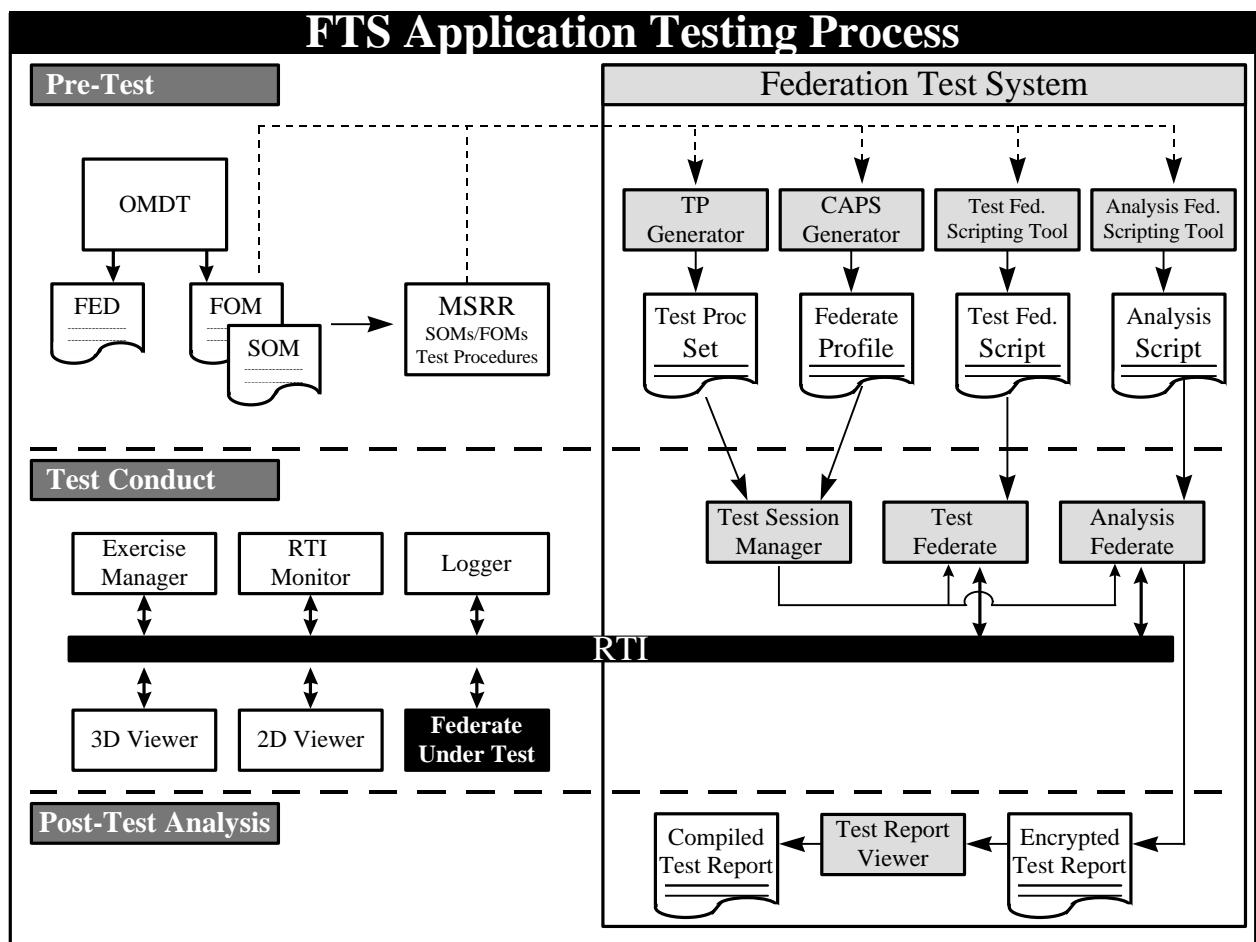


Figure 6.2. FTS Support of Application Testing

The FTS software tools are designed to be flexible and maximize reuse. Common DIFs, such as the Object Model Template DIF (OMT-DIF), are used whenever possible. The goal of the FTS design is to support all HLA Federations

and the wide range of FOM representations they present. The following sections describe the FTS software tools.

6.3.1 Test Federate

The Test Federate is used to stimulate the FTS as required by the Test Procedures to perform Federation Scenario interactions. To maximize flexibility and reuse, the Test Federate is implemented as a script reading tool. During a Test Procedure, the Test Federate will execute scripts created by the Test Federate Scripting Tool when cued by the Test Session Manager. Test Federate Scripts contain RTI service call functions, such as “Publish Attribute” and “Update Attribute Value.” These scripted functions are interpreted by the Test Federate and communicated to the RTI in accordance with the HLA Interface Specification.

6.3.2 Analysis Federate

The Analysis Federate is used to capture, analyze and report test data. The Analysis Federate is similar to the Test Federate in that it is implemented as a script reading tool. During a Test Procedure, the Analysis Federate will execute scripts created by the Analysis Federate Scripting Tool when cued by the Test Session Manager. Analysis results are reported to the tester in real-time and then written to an Encrypted Test Report file. This Encrypted Test Report file can be viewed during the Post Test Analysis phase with the Test Report Viewer tool.

6.3.3 Test Session Manager

The Test Session Manager is used to orchestrate the execution of Test Procedures created by the Test Procedure Generator. During a Test Session the Test Session Manager will:

- Provide a selection of Test Procedures to be performed.
- Provide instructions for the FUT operator, walking them through the Test Procedure scenario.
- Cue the Test Federate to execute Test Federate Scripts.
- Cue the Analysis Federate to execute Analysis Scripts.

The Test Session Manager user interface is shown in Figure 6.3.

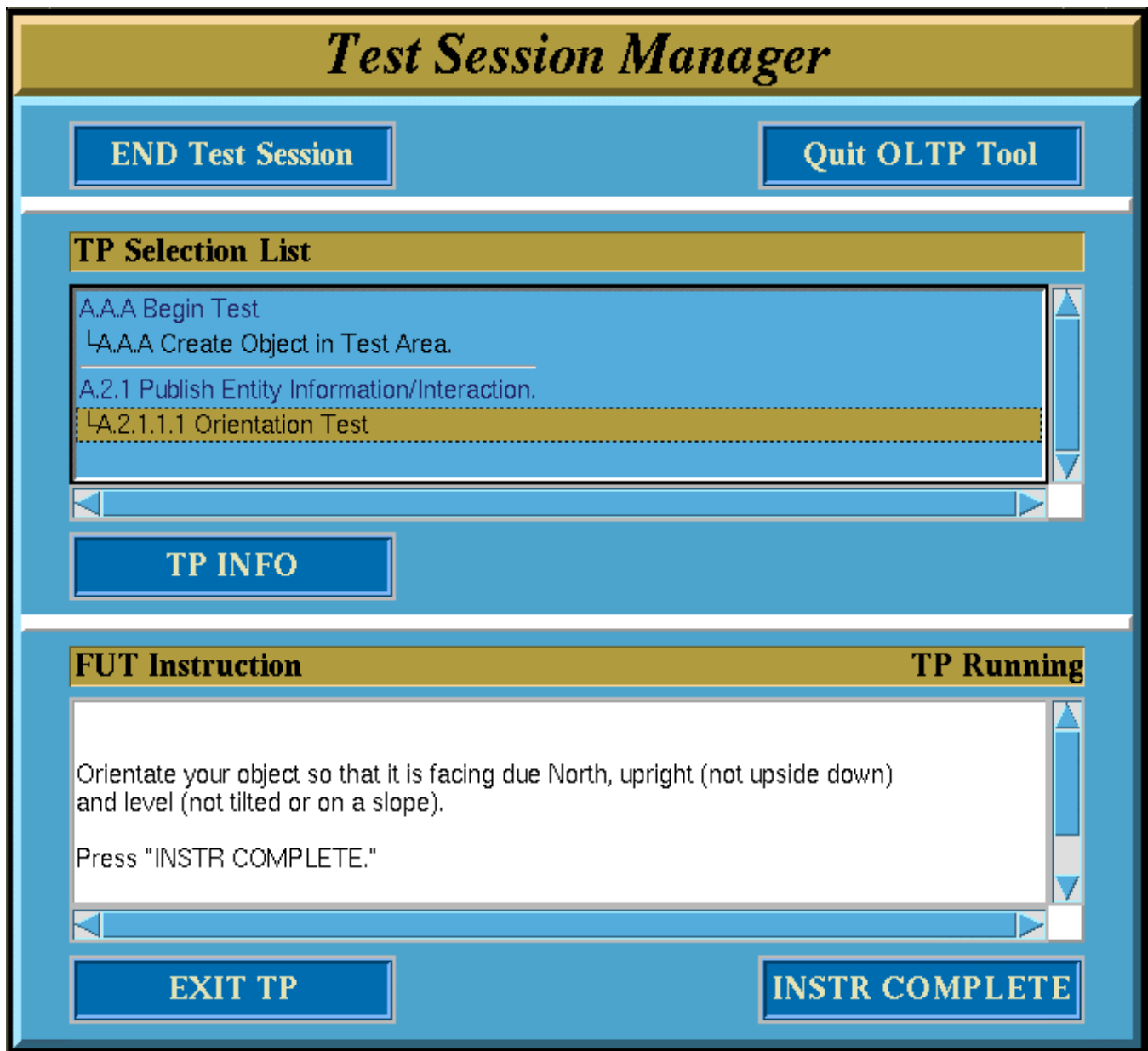


Figure 6.3. Test Session Manager

During the Pre Test phase, the intrinsic capabilities of the Federate Under Test (FUT) are captured in a Federate Profile by the CAPS Generator tool. These capabilities are used by the Test Session Manager to determine which tests need to be performed and what capabilities are to be tested.

6.3.4 Exercise Support Tools

Outside of the FTS, different exercise support tools can be useful during Application Testing to provide various perspectives of the simulation. Examples

of exercise support tools that should be widely available within the SISO and DoD communities would be, Data Loggers, Exercise Managers and RTI Monitors. Other exercise support tools, such as 2D Plan View Display Viewers and 3D Stealth Viewers, are often designed to be FOM dependent and may be limited to use within a single Federation.

6.3.5 Next Steps for the FTS

This section described the capabilities of the Federation Test System (FTS). Further development of the FTS will provide process and software tool solutions for other phases of the Federation Testing Process. See Section 7.3 for recommendations for FTS next steps.

This Page Intentionally Left Blank

7. RECOMMENDATIONS

This section provides recommendations based upon the research and development described in this report. The recommendations are divided into policy, technical, and FTS recommendation sections.

7.1 *POLICY RECOMMENDATIONS*

This section provides recommendations to STRICOM on the approach that should be taken to address the issues that have been highlighted in this document.

7.1.1 **Support Development of a DIS Federation**

STRICOM should take an active lead in the formation of a DIS Federation to provide the framework for interoperability for the many existing legacy DIS simulations.

In their Spring 1997 Simulation Interoperability Workshop (SIW) paper, "Implementation of the High Level Architecture Into DIS-Based Legacy Simulations," Braudaway and Harkrider predicted the emergence of a set of Distributed Interactive Simulation Federation Object Model (DIS FOM) standards that will "reduce the re-engineering effort and maximize the reuse of modified DIS systems in many platform oriented federations." (Braudaway & Harkrider, 1997) Alternately, perhaps a more direct approach to supporting legacy DIS simulations is the development of one community-supported, government-sponsored DIS FOM that will meet the needs of the "legacy" DIS community. This approach would allow existing DIS implementations to maximize reuse of their already-committed development efforts. The creation of a DIS FOM would offer a baseline from which DIS-based federations could add or change standard functionality as needed for their federation. As described in Section 5.1.1.2, efforts are already underway to develop a DIS-based Reference FOM, called the Real-Time Platform Reference FOM (RPR FOM).

The most urgent need for the legacy DIS community is the official formation of a DIS federation to foster interoperability between platform-level simulations, similar to the goals of DIS. This effort should be funded by users who have the most to gain from a DIS federation, such as current simulation users who have implemented DIS successfully and can benefit from the new capabilities offered by HLA (e.g., smaller messages, update upon demand, lower bandwidth common infrastructure, and federation planning tools). STRICOM, in its role as service sponsor for DIS, should support the DIS FOM effort.

7.1.2 Continue Support of the SIW Testing Forum

STRICOM should continue support of the SISO SIW Testing Forum, because it is the only organization that is addressing the technical issues involved in distributed simulation testing in an open forum.

As has been discussed, distributed simulation testing is difficult, but manageable. In order for heterogeneous distributed training systems to become increasingly accessible to the warfighter, the process needed to develop these systems must be well understood by the military user and their support contractors. They also need to understand the tools that are available to support the construction and maintenance of distributed simulation systems. The open forum represented by the SIW Testing Forum is necessary to allow discussion of these issues and provide a way for research results to be published in conference proceedings.

7.2 TECHNICAL RECOMMENDATIONS

This section provides technical recommendations based on the research and development discussed in this report. The target of these recommendations is federation and federate developers and distributed simulation testers.

7.2.1 Encourage Expansion of the FEDEP to Describe Federation Testing in More Detail

The SIW Testing Forum should push for and support the expansion of the FEDEP to include more details for federation testing.

The FEDEP description of Federation Testing does not have enough detail to provide guidance to federation developers. The current description is even misleading; by skipping steps, it makes HLA Federation Testing seem easier than it might actually be. For example, the FEDEP has no explicit descriptions of the Application Testing Phase described in this document. This implies that once federates have completed HLA Compliance Testing, they are immediately ready for Integration Testing. An explicit Application Test Phase is required before Integration to verify that individual federates are using the FOM objects, interactions, and timing protocols correctly in order to interoperate with the federation.

7.2.2 Promote the Development and Use of Interchange Formats for HLA Data

STRICOM, DMSO, SISO, Federation Sponsors, and Federate Developers should promote the development and use of interchange formats for HLA supporting and management data.

The following paragraphs discuss several data requirements in this context.

7.2.2.1 FPW

As discussed in Section 5.1.4, FPW data will support the specification of a federation execution. This data can be used in Application Testing to verify that federates are performing the services and using the objects that they have been listed as supporting as well as meeting the specified timing requirements. The FPW will be useful in Integration Testing as a catalog of the federates and their capabilities. The FPW will also be useful in Scenario Testing where automated testing tools can be used to verify that a federate is fulfilling the role defined for the federation (number of object instantiations of each object type).

7.2.2.2 MSCs for OIPs

The use of MSCs as a format for OIPs was discussed in Section 5.1.2. OIPs are required to provide a needed level of specification for the description of distributed simulation behavior. In order for OIPs to be manageable and detailed enough to be used in automated testing, they must be specified using a defined format. The MSC is proposed as one way of describing OIPs. MSCs are already being used in the development of Compliance Testing tools.

7.2.2.3 Test Procedures

Section 5.2.3 described the requirement for the use of test procedures and the development of a test procedure format to support federation testing. The idea is that test procedures can be defined in a way so that they can be used in the automation of Federation Testing. This has been demonstrated with both the DTS (for DIS compliance) and with the FTS (for HLA Application Testing.) If a format for test procedures is adopted by the community, then test procedures developed to support a federation can be stored along with the FOM in the MSRR. This approach to reusing test procedures will go a long way towards reducing the costs and effort required to perform detailed federation testing.

7.2.2.4 Federation Agreements

Federation Agreements were described in Section 5.1.3. This is the most general of the data requirements for HLA Testing, and therefore needs more definition before they can be used effectively for automated testing. Some experimentation is needed using several federations to determine what type of federation agreements are needed. These agreements should be stored in either a generic database system or one designed specifically to store requirements. Once these experiments have been conducted, then recommendations can be made on a specific approach.

7.2.3 Use the Internet to Facilitate Test Management

STRICOM and test tool developers should use internet technology as an enabler to facilitate the success of Federation Test and Management approaches.

As described in Section 5.2.5, the internet can be a useful tool in supporting Federation Testing. As networking technology continues to improve, the developers of all computer applications can assume connectivity to the internet. With this view of the future of ubiquitous networked computing, forward thinking organizations can start now to develop distributed software applications.

Federation Testing tools should be developed to support not only distributed simulations, but also distributed management, software development, testing, integration, and simulation execution.

7.2.4 Encourage Applied Research to Investigate Functional and Scenario Testing Requirements

STRICOM should encourage additional applied research efforts to understand the requirements for Functional and Scenario Testing.

The current state of the art for Functional and Scenario Testing is for Subject Matter Experts (SMEs) to watch the behavior of entities on tactical displays to judge whether they are performing correctly. Obviously, these SMEs cannot be expected to verify every critical behavior in a large federation execution. Testing tools are needed to support this process so that the more detailed and mundane tests can be carried out in an automated way, allowing the SMEs to concentrate on observable critical behavior. Additional research efforts need to focus on requirements for automated testing in these later phases of Federation Testing.

7.2.5 Encourage Basic Research to Develop Approaches for the End-to-End Federation Testing Automation

STRICOM should encourage basic research addressing the feasibility of end-to-end automation of the Federation Testing Process.

The concept of the Federation Test Phases making up an automation hierarchy of data and method relationships was presented briefly in Section 4.2.6. This approach to specifying, implementing, and testing distributed simulation systems offers the potential for drastic cost reductions in testing time and resources. In addition, this level of automation will be critical for the rapid development of distributed simulation systems using off-the-shelf simulation components.

7.2.6 Promote the Practice of Providing Test Procedures with Standards Proposals

The SIW Testing Forum should actively promote the requirement of providing test procedures with SISO Standards Proposals.

As has been discussed in the Testing Forum, better standards can be developed if the standards development groups are thinking about the testability of the product. Fortunately, the HLA standards will be forwarded with test procedures. If the RPR FOM proposal is forwarded as an HLA Draft Standard, community effort will have to go towards converting the DIS Compliance Test Procedures into test procedures supporting the HLA RPR FOM implementation.

7.2.7 Leverage Testing Research and Practice

Military distributed simulation developers and testers need to make use of existing research, methods, and tools that have been developed to support software engineering, software testing, distributed systems development, real-time software testing, Test and Evaluation, and distributed simulation testing.

The recommendation here is that simulation professionals need to take advantage of the technical contributions of previous programs, published research, and tools that have been developed to address some of their issues. Specifically, software engineering research and methods in testing and especially distributed software and hardware testing need to be leveraged more fully. The opportunity exists for the integration of automated testing tools developed in the general software community to be used to do distributed simulation testing. Regression testing tools are an immediate opportunity.

7.3 FTS DEVELOPMENT RECOMMENDATIONS

As discussed in Section 5.3, a Federation Testing Process has been proposed that contain four phases: Application Testing, Integration Testing, Functional Testing, and Scenario Testing. The Federation Testing System (FTS) was designed to support this process, and is currently able to directly support Application Testing and some aspects of Integration Testing.

The next recommended steps for the FTS are to test it with federates and other tools, and to continue tool integration and development to support Integration Testing fully and to support the latter phases of Federation Testing (Functional and Scenario Testing.)

7.3.1 Use the FTS to Support Federate and Tool Development

The FTS needs to be exposed to HLA users as much as possible in order to get the feedback necessary to improve the system. The focus of this alpha/beta testing effort should be on the federate/tool developer and the federation integrator.

One of the persistent problems in the development of automated test tools is the need to "test" the test tool (Clay et al., 1995). FTS currently is undergoing alpha testing with federates that have implemented the RPR FOM. It will be important in these early tests to understand that the use of the RPR FOM to achieve interoperability with HLA is still in its infancy. FTS should not be advertised as a finished tool, but as one of several early adopters of the HLA and, specifically, the RPR FOM.

Both HLA federate and tool developers need to interoperate with other federates during their development and unit testing efforts. FTS should be offered to those developers who see the need for a flexible test federate to develop their unit testing stubs (as discussed in an earlier section.)

Several tools and federates have been identified as potential alpha testers for FTS. Actually, all of the tools involved in the process will be testing each other's implementations -- essentially bringing each other up to a higher implementation confidence level. One of the early implementations used already with the FTS is the IST Gateway (Wood, Petty, Cox, Hofer, & Harkrider, 1997).

The FTS team should take advantage of the opportunity to be certified as HLA compliant by participating in the HLA Conformance Testing Process described in Section 4.1.2.

During the early rollout of the FTS, it will be important to create a lab environment that will promote learning and tool capability improvement. Model lab environments currently exist at the TASC development site, at STRICOM, and in GTRI's Distributed Simulation Systems (DSS) Lab. (The DSS Lab is described in Section 9.4)

After undergoing initial alpha tests with other early adopters of the RPR FOM, additional effort is required in several aspects of the FTS, including:

- Expanding support of the RPR FOM to include comprehensive testing of all objects and interactions (this supports the standing-up of a DIS federation).
- Expanding support of non-DIS-related federations (this proves FTS value outside of supporting DIS-legacy systems).

- Participating in the early phase of federate development (supporting unit testing).
- Participating in the early phases of federation development (supporting Application Testing).

7.3.2 Use the FTS to Support Federation Development

The FTS needs to be used to support development of HLA federations.

Several near-term opportunities exist for developing lessons learned in a federation development effort, including support of Cameleon, JSIMS, and JADS. For these efforts, lessons learned in Integration Testing, Functional, and Scenario Testing will be used to improve the FTS, as well as to help identify the use of other tools needed in the later phases of Federation Testing. These lessons learned also will help to improve the data standards needed for Federation Testing, including Test Procedures, Federation Agreements, and Object Interaction Protocols (OIPs).

7.3.3 Develop Integration, Functional, and Scenario Testing Capabilities

The FTS should be expanded to support the needs of the latter phases of the Federation Testing Process (Integration, Functional, and Scenario Testing).

This goal will end up being a long-term effort and will rely on further research on Functional and Scenario Testing, as well as the formation of more HLA federations.

7.4 TIMELINE

This section introduces a timeline (Figure 7.1) that shows graphically the events leading up to and including the current state of the FTS.

This timeline also projects future development activities that have not yet been funded, but are needed to support Integration, Functional, and Scenario Testing.

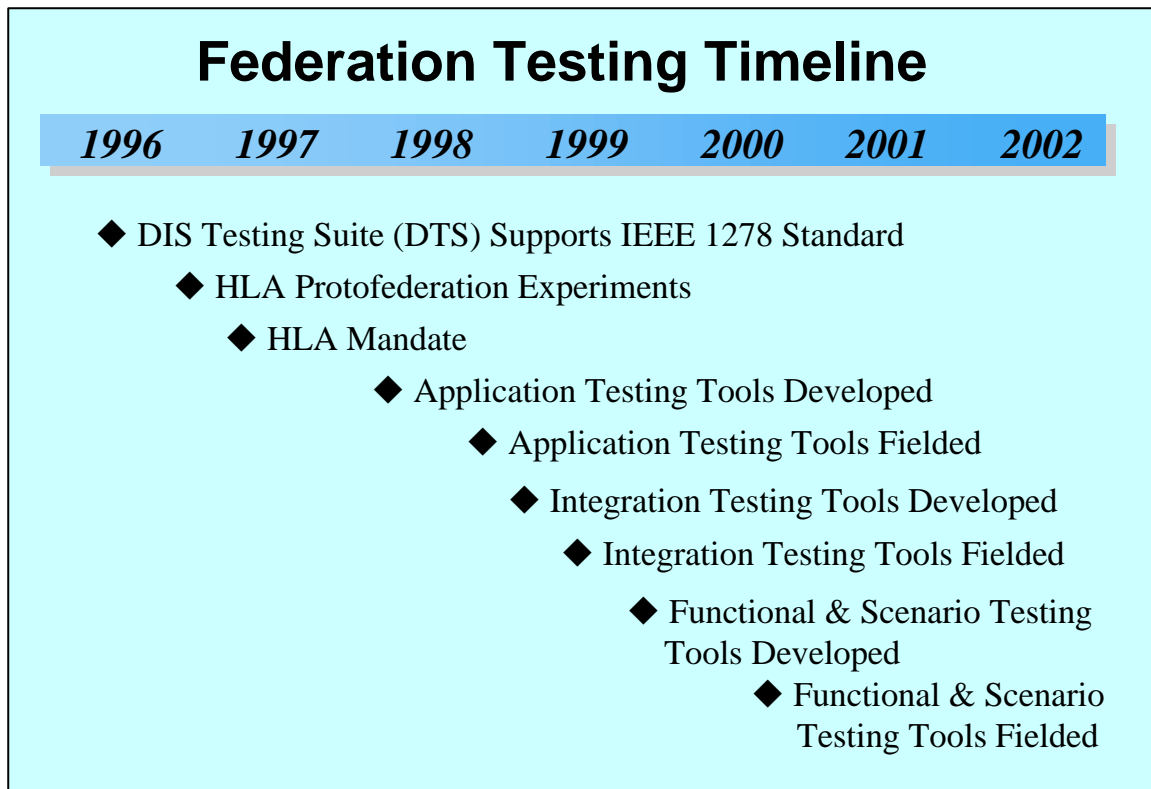


Figure 7.1. Federation Testing Development Timeline

8. REFERENCES

Acusoft. (1997a). Federation Test System (FTS) Application Testing Process Requirements Definition (Contract N61339-95-D-0006). Orlando, FL: STRICOM.

Acusoft. (1997b). Federation Test System (FTS) Application Testing Process System/Subsystem Design Description (S/SDD) (Contract N61339-95-D-0006). Orlando, FL: STRICOM.

Braudaway, W. K., & Harkrider, S. M. (1997, March 3-7). Implementation of the High Level Architecture into DIS-Based Legacy Simulations. Paper presented at the 1997 Spring Simulation Interoperability Workshop, Orlando, FL.

Clay, B., Roberts, D. W., & Stueve, J. (1995, March 13-17). Real Time Distributed Simulation Integration Testing. Paper presented at the 12th Workshop on Standards for the Interoperability of Defense Simulations, Orlando, FL.

Dahmann, D. J. (1998, March 9-13). Persistent Federations. Paper presented at the 1998 Spring Simulation Interoperability Workshop, Orlando, FL.

Dahmann, J. S., Olszewski, J., Briggs, R., Richardson, R., Weatherly, R. M., Calvin, J., & Zimmerman, P. (1997, September 8 - 12). High Level Architecture (HLA) Performance Framework. Paper presented at the 1997 Fall Simulation Interoperability Workshop, Orlando, FL.

DMSO. (1996a). DoD VV&A Recommended Practices Guide, <<http://www.dmsomil/docslib/mspolicy/vva/rpg/>> .

DMSO. (1996b). FED Process Boxes, <<http://hla.dmsomil/hla/federation/fedep/fed/boxes.html>> : DMSO.

DMSO. (1996c). High Level Architecture Federation Development and Execution Process (FEDEP) Model, Version 1.0 . Alexandria, VA: Defense Modeling and Simulation Office.

DMSO. (1996d). High Level Architecture Management Object Model, Version 0.2 : Defense Modeling and Simulation Office.

DMSO. (1997a). Federate Test Tools Operator Guide, Version 1.1. Paper presented at the 1997 Fall Simulation Interoperability Workshop, Orlando, FL.

DMSO. (1997b). High Level Architecture Federation Development and Execution Process (FEDEP) Model, Version 1.1 . Alexandria, VA: Defense Modeling and Simulation Office.

DMSO. (1997c). High Level Architecture Interface Specification, Version 1.1 .

DMSO. (1997d). High Level Architecture Object Model Template Data Interchange Format, Version 1.1 . Alexandria, VA: Defense Modeling and Simulation Office.

DMSO. (1997e). High Level Architecture Rules, Version 1.2 : Defense Modeling and Simulation Office.

DoD. (1996). DoD Instruction 5000.61,"DoD Modeling and Simulation (M&S) Verification, Validation, and Accreditation (VV&A), <http://www.dmsomil/docslib/mspolicy/vva/dodifin.doc> .

Graffagnini, J. (1997, March 3-7). An Automated Approach to HLA Testing. Paper presented at the 1997 Spring Simulation Interoperability Workshop, Orlando, FL.

Hansen, S. (1994, March 14-18). War Breaker DIS Integration & Testing -- Past ...Present...Future. Paper presented at the 10th Workshop on Standards for the Interoperability of Defense Simulations, Orlando, FL.

Hyon, T. C., & Seidel, D. W. (1997, March 3-7). Design and Implementation of High Level Architecture (HLA) Management Object Model (MOM) in the RTI Version F.0. Paper presented at the 1997 Spring Simulation Interoperability Workshop, Orlando, FL.

IEEE. (1993). IEEE Standard for Information Technology -- Protocols of Distributed Interactive Simulation Applications, Version 1.0 (IEEE Std 1278-1993). New York.

IEEE. (1997a). IEEE Home Page, <http://hla.dmsomil/hla/federation/fedep/fed/boxes.html> .

IEEE. (1997b). IEEE Recommended Practice for Distributed Interactive Simulation--Verification, Validation, and Accreditation (IEEE Std 1278.4-1997). New York.

Kanewske, C., & Fine, S. (1998, March 9-13). STOW Systems Integration, Tools and Applications. Paper presented at the Spring '98 Simulation Interoperability Workshop, Orlando, FL.

Knightson, K. (1993). OSI Protocol Conformance Testing: IS 9646 Explained. New York: McGraw-Hill.

Law, A. M., & Kelton, W. D. (1991). Building Valid and Credible Simulation Models, Simulation Modeling and Analysis (2nd Edition ed., pp. pp. 298-324). New York: McGraw-Hill, Inc.

Lewis, R. (1997, March). A Generic Model for Verification, Validation, and Accreditation (VV&A) of Advanced Distributed Simulations (ADS) Used for Test and Evaluation (T&E). Paper presented at the the 1997 Spring Simulation Interoperability Workshop, Orlando, FL.

Long, R., McAuliffe, M., & Liu, J. (1996, March 11-15). DIS Capabilities Statement. Paper presented at the 14th Workshop on Standards for the Interoperability of Defense Simulations, Orlando, FL.

Loper, M. L. (1993, March 22-26). Recommendations for DIS Based on Testing Performed at the I/ITSEC Demonstrations. Paper presented at the The Eighth Workshop on Standards for the Interoperability of Defense Simulations, San Antonio, TX.

Loper, M. L., Goldiez, B., & Smith, S. (1993a, November 29 - December 2). The 1992 I/ITSEC Distributed Interactive Simulation Interoperability Demonstration. Paper presented at the 15th Interservice/Industry Training Systems and Education Conference, Orlando, FL.

Loper, M. L., Goldiez, B., Smith, S., Ng, H., Craft, M., Petty, M., Bulumulle, G., & Lisle, C. (1993b). I/ITSEC DIS Interoperability Demonstration Test Procedures and Results (IST-TR-93-04). Orlando, FL: Institute for Simulation and Training.

McAuliffe, M., Long, R., Liu, J., Hays, J., & Nocera, D. (1995, March 13-17). Testing Applications for DIS. Paper presented at the 12th Workshop on Standards for the Interoperability of Defense Simulations, Orlando, FL.

McAuliffe, M., Long, R., Liu, J., Hays, J., & Nocera, D. (1996, March 11-15). DIS Test Suite (DTS). Paper presented at the 14th Workshop on Standards for the Interoperability of Defense Simulations, Orlando, FL.

McKee, D. L. (1998, March 9-13). Compensating for Latency Variations in Air-to-Air Missile T&E Using Live Aircraft Linked to a Missile HWIL Simulation. Paper presented at the 1998 Spring Simulation Interoperability Workshop, Orlando, FL.

McLean, T., & Loper, M. (1997). Management Object Model Extension to Support Service Logging in Federation Compliance Testing (See DMSO web page < <http://hla.dmsomil.gov> >). Alexandria, VA: Georgia Institute of Technology in Support of DMSO.

Morley, D. S., & McLean, T. (1995, September 18-22). Issue Management Database System. Paper presented at the 13th Workshop on Standards for the Interoperability of Defense Simulations, Orlando, FL.

Page, E. H., & Babineau, W. E. (1997, September 8-12). The ALSP Joint Training Confederation: A Case Study of Federation Testing. Paper presented at the 1997 Fall Simulation Interoperability Workshop, Orlando, FL.

Roberts, D. W. (1995, April). Integration Test of Distributed Interactive Simulation Exercises. Paper presented at the SPIE International Symposium - Aerospace and Defense Electronic Systems Technical Conference, Orlando, FL.

Roberts, D. W., Collins, T. R., Esslinger, D. I., Horst, M. M., Johnson, T. B., Marks, J. R., McLean, A. L. M. T., & Wallace, J. C. (1997a). High Level Architecture Simulation Interface Phase One Final Technical Report (GTRI-TR-97-E8604-102F). Atlanta, GA: Georgia Institute of Technology (GTRI), Georgia Institute of Technology.

Roberts, D. W., Horst, M., Old, J. A., Lashley, T., Freeman, R., Mullally, K., Long, R., & Gajkowski, B. J. (1997b, March). The HLA Federation Testing Process (Presentation). Paper presented at the 1997 Fall Simulation Interoperability Workshop, Orlando, FL.

Schach, S. R. (1990). Testing, Software Engineering (pp. pp. 110-168). Homewood, IL: Richard D. Irwin, Inc. and Aksen Associates, Inc.

Scudder, R. O., & Sheehan, J. H. (1997, March 3-7). HLA FOM/SOM Content Standards. Paper presented at the 1997 Spring Simulation Interoperability Workshop, Orlando, FL.

Shen, D. T., Loper, M. L., & Ng, H. K. (1992, November 2-4). DIS Application Protocol Testing Using a Formal Description Technique. Paper presented at the 14th Interservice/Industry Training Systems and Education Conference, Orlando, FL.

Vanzant-Hodge, A. (1994, March 14-18). IST Test Tools (Presentation in the DIS Testing SIG). Paper presented at the 10th Workshop on the Standards for the Interoperability of Defense Simulations, Orlando, FL.

Vanzant-Hodge, A., Cheung, S., & Smith, S. (1994, November 28 - December 1). Testing Conformance for Distributed Interactive Simulation (DIS) Standards. Paper presented at the 16th Interservice/Industry Training Systems and Education Conference, Orlando, FL.

White, E. (1998, March 9-13). A Conceptual Model for Simulation Load Balancing. Paper presented at the 1998 Spring Simulation Interoperability Workshop, Orlando, FL.

Wood, D. D., Petty, M. D., Cox, A., Hofer, R., & Harkrider, S. M. (1997, March 3-7). HLA Gateway Status and Future Plans. Paper presented at the 1997 Spring Simulation Interoperability Workshop.

Wuerfel, R. (1998, March 9-13). A Comparison of HLA and DIS Real-Time Performance. Paper presented at the 1998 Spring Simulation Interoperability Workshop, Orlando, FL.

Zimmerman, P. M., & Harkrider, S. (1997, September). FEDEP Phase III Examination: Federation Testing, Execution, and Feedback. Paper presented at the 1997 Fall Simulation Interoperability Workshop, Orlando, FL.

This Page Intentionally Left Blank

9. APPENDIX

9.1 ModSAF RDR

This section provides the details of the ModSAF libreader formats described in Section 5.2.2.1.

Integer	1 2 3 1634 0x78ffff 0xFF78FE 0X78ffff 0o446 00664
Real	2.0 35.67 1.634e+3
Symbol	foo bar67 baz-foo foo_bar "a+b" "1.0" 0o8000
Array	(1 2 3) (1.1 2 3.1 bar67) (tank (weapons 25mm) (weight 25.0)

Table 9.1. ModSAF libreader Data Items

```
typedef union reader_union {
char          *charptr;
int32         integer;
float32       real;
union reader_union *array;
} READER_UNION;
```

Table 9.2. ModSAF Data Structure Format

```
;; $RCSfile$ $Revision$ $State$
;;
;; This file specifies information about how routemap should calculate
;; corridors for various terrain databases.  The "default"
;; values are used if the database name is not found.
;;
;; Also in this file are the soil types which are considered obstacles.
;; The format is as follows:
;; (
;;   (databases
;;     (<database name> (max_error <maximum deviation from the actual
data>)
;;
;;                       (max_corridor <maximum corridor length>)
;;
;;                       )
;;   ...
;; )
;; (soils
```

```

;; (boulders <boulder soil> <boulder soil> ...)
;; (lakes <lake soil> <lake soil> ...)
;; )
;; )
(
  (databases ("default" (max_error 50.0)
                    (max_corridor 1000.0)
                    (max_grow 0.2)
                    (max_shrink 0.33)
                    (min_expansion_dist 50.0)
                    (add_point_distance 0.0)
                    (add_point_width_ratio 0.0)
                  )
    ("knox" (max_error 25.0)
              (max_corridor 500.0)
              (max_grow 0.2)
              (max_shrink 0.33)
              (min_expansion_dist 50.0)
              (add_point_distance 0.0)
              (add_point_width_ratio 0.0)
            )
    ("hunter" (max_error 50.0)
               (max_corridor 1000.0)
               (max_grow 0.2)
               (max_shrink 0.33)
               (min_expansion_dist 50.0)
               (add_point_distance 0.0)
               (add_point_width_ratio 0.0)
             )
  )
;; (soils (lakes SOIL_DEEP_WATER)
;;       (boulders SOIL_NO_GO)
;;       )
;; We can't use macros here because they are defined after the routemap
is
;; created
(soils (lakes 4)

```

```
(boulders 15)  
)  
)
```

Table 9.3. Sample ModSAF .rdr file:

9.2 MESSAGE SEQUENCE CHART (MSC) DEPICTING A FIRE AND DETONATE SERIES BETWEEN TWO FEDERATES

This section provides a detailed example of the MSC textual format described in Section 5.1.2.

```
MSCDOCUMENT ;

MSC ;

INSTANCE Federate 1 ;

    OUT Publish Object Class (MilitaryPlatformEntity, Position,
DamageState,...) TO RTI ;

    OUT Subscribe Object Class Attribute (MilitaryPlatformEntity,
Position, DamageState...) TO RTI ;

    OUT Update Attribute Values (MilitaryPlatformEntity1, Position,
DamageState,...) TO RTI ;

    IN Reflect Attribute Values (MilitaryPlatformEntity2, Position,
DamageState,...) FROM RTI ;

    OUT Publish Interaction Class (WeaponFire) TO RTI ;

    OUT Publish Interaction Class (Munition Detonation) TO RTI ;

    OUT Send Interaction (WeaponFire, EventID, FiringLocation,...) TO RTI ;

    OUT Send Interaction (Munition Detonation, EventID, FiringLocation,...)
TO RTI ;

    IN Receive Attribute Values (MilitaryPlatformEntity2, Position,
DamageState,...) FROM RTI ;

ENDINSTANCE;

INSTANCE RTI ;
```

```

CONCURRENT ;

    IN Publish Object Class (MilitaryPlatformEntity, Position,
DamageState,...) FROM Federate 1 ;

    IN Subscribe Object Class Attribute (MilitaryPlatformEntity,
Position, DamageState,...) FROM Federate 2 ;

    IN Subscribe Object Class Attribute (MilitaryPlatformEntity,
Position, DamageState,...) FROM Federate 1 ;

    IN Publish Object Class (MilitaryPlatformEntity, Position,
DamageState...}) FROM Federate 2 ;

ENDCONCURRENT ;

•
•
•

    OUT Receive Interaction (Munition Detonation, EventID,
FiringLocation,...) TO Federate 2 ;

    IN Update Attribute Values (MilitaryPlatformEntity2, Position,
DamageState,...) FROM Federate 2 ;

    OUT Receive Attribute Values (MilitaryPlatformEntity2, Position,
DamageState,...) TO Federate 1 ;

ENDINSTANCE;

INSTANCE Federate 2 ;

OUT Subscribe Object Class Attribute (MilitaryPlatformEntity, Position,
DamageState,...) TO RTI ;

•
•
•

ACTION Process
    Detonation
    Effects

```

```
OUT Update Attribute Values (MilitaryPlatformEntity2, Position,  
DamageState,...}) TO RTI ;
```

```
ENDINSTANCE;
```

```
ENDMSC;
```

```
ENDMSCDOCUMENT;
```

Table 9.4. Abridged Textual Representation of a Message Sequence Chart (MSC) Depicting a Fire and Detonate Series between Two Federates

9.3 FTS TEST PROCEDURES EXAMPLE

This section provides a detailed example of the Test Procedure format described in Section 5.2.3.1.

```
<TP> 1.1 Update Attributes: 00001 1.1.1 Update PhysicalEntity
Appearance Attribute.

<INIT_COND>

The FTS will subscribe to all of the attributes in the PhysicalEntity
class.

The FUT will publish all of attributes in the PhysicalEntity class that
it is capable of updating.

</INIT_COND>

<REQS>

Each Federate shall update the their appearances attributes in
accordance with PhysicalEntity class of the RPR FOM Version 0.1.5.

</REQS>

<CAPS>

The Federate is capable of updating their appearances attributes in the
PhysicalEntity class.

</CAPS>

<SCENARIO>

The FUT will update the appearance attributes in the PhysicalEntity
class that it is capable of updating.

The FTS will analyze the updated attributes and ensure that the
attribute values correspond to the RPR FOM V0.1.5.

</SCENARIO>

<CRITERIA>

PASS: FUT correctly updated PhysicalEntity appearance attributes in
accordance with the RPR ROM V0.1.5.

FAIL: FUT incorrectly updated PhysicalEntity appearance attributes in
accordance with the RPR ROM V0.1.5.

</CRITERIA>

<PAGE>

<TITLE>

Update PhysicalEntity Appearance Attribute. - Page 1

</TITLE>

<FUT_INSTR>

Ensure that your simulation is publishing the PhysicalEntity Attributes
that it is capable of updating.
```

```

Press INSTR COMPLETE.
</FUT_INSTR>
</PAGE>
<PAGE>
<TITLE>
Update PhysicalEntity Appearance Attribute. - Page 2
</TITLE>
<FUT_INSTR>
Manipulate your simulation to update all of the appearance attributes
in the PhysicalEntity class that it is capable of updating.

Press INSTR COMPLETE.
</FUT_INSTR>
<ANALYSIS_CUE>
1_1_1_AFscript1.afs
</ANALYSIS_CUE>
</PAGE>
</TP>

```

Table 9.4. FTS Test Procedure Example

9.4 DISTRIBUTED SIMULATION SYSTEMS (DSS) LAB

This section describes the DSS Lab. This example of a lab environment should be useful for federation developers who need to create a working environment for development and testing.

9.4.1 DSS Lab Environment Setup for Federation Tests

One of the first priorities in forming the Distributed Simulations Systems (DSS) group at GTRI was to develop a lab to support our research projects. The lab supports several different platforms from Silicon Graphics and SUN to Windows NT. The idea is to have access to an environment for such things as software testing (including the Runtime Infrastructure (RTI)) and developing and/or supporting a federation. Our lab also is an ideal environment to test ideas and support Federation Testing.

9.4.1.1 Multiple RTI Versions

The RTI has been in use in the DSS lab since the first release became available. We have followed its progress, installing and using each of the new versions as they were released, and using these new versions to upgrade our test tools, the Georgia Tech protoFederation (GTpF), the HLA Distributed Simulation Interface Framework (DSIF), and several simple test federates. The following versions of the RTI are installed in the DSS lab:

- RTI F.0 - "familiarization" release that implemented a subset of Interface Specifications (IFSpec) 1.1
- RTI 1.0.1 - the first official release of the RTI for IFSpec. 1.1
- RTI 1.0.2 - fixed memory and requestLBTS bugs
- RTI 1.0.3 - fixed MOM-related bugs and an NT time-management bug
- RTI 1.3 - all HLA Interface Spec 1.3 services, Sun/Solaris 2.4

Because of our involvement in defining the Management Object Model (MOM), we also have had access to intermediate, developmental versions, which we have installed and used to implement and test the HLA Compliance Testing tools. These versions included increments between 1.0.1 and 1.0.2, and between 1.0.2 and 1.0.3. They also include preliminary versions of the RTI in development for IFSpec 1.3.

9.4.1.2 DIS to HLA Migration

As part of our commitment to staying informed about the latest developments in the distributed simulation domain, we stay abreast of the latest releases of all different types of software, including both ModSAF 3.0 and the Distributed Interactive Simulation (DIS) Test Suite (DTS). The DSS lab is in position to support an HLA version of ModSAF and the Federation Testing System (FTS). Also, because our lab has access to tools such as the Aegis OMDT, it is an ideal environment to support Federation Testing.

9.4.1.3 JMASS and Threat Radar Simulation Federates Used in IRAD Project

The Georgia Tech protoFederation (GTpF) was developed for the HLA Simulation Interface Internal Research and Development (IRAD) Project, which focused on developing an understanding of how legacy simulations should be adapted for HLA use. This project is discussed in detail in the "High Level Architecture Simulation Interface, Phase One Final Technical Report" (Roberts et al., 1997a). The DMSO HLA Federation Development Process (FEDEP) (DMSO, 1996c) was used as a guideline for development. A survey of GTRI simulations was conducted to identify candidate simulations for an HLA federation. From this survey, two legacy simulations were selected and modified to interoperate using the RTI. The simulations used were the Target Engagement Radar (TER) simulator, developed in GTRI's System Development Laboratory (SDL), and the radar propagation and target backscatter models from the Joint Modeling and Simulation System (JMASS), exercised in GTRI's Electronic Systems Laboratory (ELSYS). In addition, the federation manager and data logging tools used in GTRI's Information Technology and Telecommunications Laboratory (ITTL) contributed to the federation.

The GTpF has been successfully integrated and executed. Since neither of the legacy simulations have graphical display capabilities for assessing the collected data, additional tools are under development to reduce and analyze the data and produce graphical displays.

Several lessons were learned from the federation development efforts, particularly in the areas of:

- guidelines for Simulation Object Model (SOM) and Federation Object Model (FOM) development
- effects of federation design on performance
- issues with federation initialization, data formats, and time management

These lessons are guiding the design and development of a simulation interface framework in Phase Two of this project.

9.4.2 HLA Distributed Simulation Interface Framework (DSIF)

One of the lessons learned from the first-year efforts of the HLA IRAD project is that creating HLA federate simulations currently is more difficult than necessary. The RTI API provides an abstraction from the communications layer of distributed simulation. However, there are many common tasks for which code must be written to use the RTI.

All federates must join and resign from the federation execution, initialize timing parameters, add and remove items from HandleValuePairSets, and convert data between transportation representation and native platform representation. Many of these tasks can be implemented once, then reused in other simulations. Code generation can tailor the other common tasks to the object model for a particular simulation.

In addition, there is room for a higher level of abstraction from the RTI API and HLA IFSpec, which is a distributed simulation paradigm that embodies common principles of distributed simulation. In this framework, the proposed paradigm is based on the concept of a shared universe of objects, which have persistent state as described by their attributes, and events (HLA interactions), which may occur but are transitory. Each simulation has knowledge of all shared objects, but is responsible only for updating the object attributes it owns. Each simulation also may add additional internal state information to its knowledge of the shared objects. The main run loop for each simulation has three steps:

- update the internal state of all objects
- synchronize with the universe (changes made in the simulation are sent out to the universe, and changes made by other simulations in the universe are received)
- increment time

The Distributed Simulation Interface Framework (DSIF) implements this distributed simulation paradigm using foundation classes and code generation. The foundation classes are intended to implement all distributed simulation tasks that are independent of an object model. They currently abstract from RTI handle mapping and HandleValuePairSets, provide RTI request and exception handling, and provide a common method for implementing data conversion for each data type. This approach minimizes the code changes required to experiment with different data representations. The foundation classes also include maskable message logging, which provides highly configurable data logging and debugging output options. The code generator implements a SOM or FOM by generating code for classes derived from the foundation classes to implement the specific object model. The user is then responsible only for providing his domain-specific object model and the behavior code for his objects.

9.4.2.1 Current Status

An alpha version of the DSIF currently is implemented in C++ for the RTI 1.0.3 on a Solaris platform. In this version, the foundation classes implement about two-thirds of the services in HLA IFSpec 1.1 and the code generator generates object and event classes from either a SOM or FOM. Alpha testing is scheduled for Spring 1998 and includes use of the DSIF with a FORTRAN legacy simulation. Future plans for the framework include:

- ports to C++ for Windows NT and to Java
- implementation of remaining IFSpec 1.1 services and evolution to IFSpec 1.3
- implementation of a common simulation configuration mechanism
- additional code generation using the Conformance Statement
- scenario generation from the Fedex Planning Workbook (FPW)
- expansion of timing model support

9.4.2.2 Relevance to Federation Testing

The DSIF and the concepts behind it are important to Federation Testing for several reasons. Using a framework to build simulations decreases the amount of time needed to test each individual simulation. Functionality provided by the framework can be tested extensively once. This functionality does not need to be re-tested in each simulation developed using the framework. The framework also will standardize input and output data file types and formats, which will, in turn, standardize test suite data. Also, using a framework can simplify usability of simulations. Once someone learns how to set up one framework-based simulation, the next framework-based simulation will be easier to use because the same types and formats of files are used, even though the number of files and their contents will be different.

9.5 ACRONYM LIST

AAI - All-Actor Integration

ADS - Advanced Distributed Simulation

ALSP - Aggregate Level Simulation Protocol

AMG - Architecture Management Group

BNF - Bacus Naur Format

C4I - Command, Control, Communications, Computers, and Intelligence

CA - Certification Agent

COO - Concept of Operation

CS - Capability Statement

CSR - Certification Summary Report

DIF - Data Interchange Format

DIS - Distributed Interactive Simulation

DMSO - Defense Modeling and Simulation Office

DSIF - Distributed Simulation Interface Framework

DSS - Distributed Simulation Systems

DTS - DIS Test Suite

ELSYS - Electronic Systems Laboratory

EMF - Exercise Management & Feedback

FEDEP - Federation Development and Execution Process

FII - Functional Interface Integration

FOM - Federation Object Model

FPW - Federation Execution (Fedex) Planning Workbook

FRED - Federation Required Execution Details

FTAM ISS - Federation Test and Management Integration Support & Study

FTS - Federation Testing System

FUT - Federate Under Test

GTpF - Georgia Tech protoFederation

HLA - High Level Architecture

IDT - Integrated Development Team

IEEE - The Institute of Electrical & Electronics Engineers

IFSpec - Interface Specification

IPT - Integrated Product Team

ITTL - Information Technology and Telecommunications Laboratory

JMASS - Joint Modeling and Simulation System

JTC - Joint Training Confederation

LHS - left-hand side

M&S - Models and Simulations

MOM - Management Object Model

MSC - Message Sequence Chart

MSRR - Model and Simulation Resource Repository

OIP - Object Interaction Protocol

OMDD - Object Model Data Dictionary

OMDT - Object Model Development Tools

OML - Object Model Library

OMT - Object Model Template

OPFOR - opposing force

PROC - Federation Development Process Forum

RepSOM - Representative SOM

RHS - right-hand side

RPR FOM - Real-Time Platform Reference FOM

RTI - RunTime Infrastructure

S/SDD - System/Subsystem Design Description

SAMs - Surface-to-air-missiles

SDL - System Development Laboratory

SISO - Simulation Interoperability Standards Organization

SIW - Simulation Interoperability Workshop

SOM - Simulation Object Model

T&E - Test and Evaluation

TER - Target Engagement Radar

USD(A&T) - Under Secretary of Defense for Acquisition and Technology

UTC - Coordinated Universal Time

VV&A - Verification, Validation, and Accreditation